



This work is protected by copyright and other intellectual property rights and duplication or sale of all or part is not permitted, except that material may be duplicated by you for research, private study, criticism/review or educational purposes. Electronic or print copies are for your own personal, non-commercial use and shall not be passed to any other individual. No quotation may be published without proper acknowledgement. For any other use, or to quote extensively from the work, permission must be obtained from the copyright holder/s.



Detection of suspicious URLs in online social networks using supervised machine learning algorithms

Mohammed Fadhil Zamil Al-Janabi

Submitted for the degree of

Doctor of Philosophy

December 2018

Keele University

TABLE OF CONTENTS

LIST OF FIGURES	VII
LIST OF TABLES	XI
GLOSSARY OF TERMS.....	XIII
ACKNOWLEDGEMENTS	XVII
ABSTRACT	XIX
CHAPTER 1 INTRODUCTION	21
1.1 Introduction	22
1.2 Background	23
1.3 Motivations.....	24
1.4 Problem statement	25
1.5 Objectives.....	25
1.6 Research questions	26
1.7 Main contributions of thesis	26
1.8 Structure of thesis.....	28
CHAPTER 2 LITERATURE REVIEW	31
2.1 Technical background	32
2.2 What is spam content?.....	33
2.2.1 Twitter’s spam rules	35
2.3 Spam mechanism.....	37
2.3.1 Sybil attacks and fake accounts.....	39
2.3.2 Fake accounts and the black market.....	40
2.3.3 What are spambots?	41
2.4 Countermeasures	42

2.4.1	Blacklist.....	42
2.4.2	Introduction to machine learning methods.....	44
2.4.3	Machine learning algorithms.....	49
2.4.4	Ensemble learning algorithms.....	54
2.5	Features used to build spam classifiers	58
2.5.1	Twitter accounts and content features.....	59
2.5.2	URL, hosting and web page features	63
2.6	Conclusion.....	67
CHAPTER 3 RESEARCH METHODOLOGY		69
3.1	Background	70
3.2	Data sources and labelling methods	73
3.2.1	Data collection process.....	74
3.2.2	Data labelling process	75
3.2.3	Data preparation and unbalance issue	81
3.3	Feature extraction and engineering	82
3.3.1	Features used in building classifiers.....	84
3.4	Model selection	96
3.4.1	Model selection criteria.....	97
3.5	Hyperparameter tuning.....	101
3.5.1	Feature selection methods	102
3.6	Discussions.....	103
CHAPTER 4 USING SUPERVISED MACHINE LEARNING ALGORITHMS		
TO DETECT SUSPICIOUS URLS IN TWITTER		106
4.1	Introduction	107

4.2	Model selection	108
4.3	Model performance enhancement	110
4.3.1	Model enhancement through parameter tuning.....	111
4.3.2	Experiment summary	117
4.3.3	Hyper-parameter tuning and overfitting.....	119
4.4	Model enhancement by feature selection	120
4.5	Enhance model performance by adding more training data.....	126
4.6	Conclusion.....	127

CHAPTER 5 MORE INFORMATIVE FEATURES AND ENSEMBLE

	LEARNING METHODS USED TO DETECT MALICIOUS URLS ON TWITTER	129
5.1	Introduction	130
5.2	New dataset and new features deployed	130
5.3	Ensemble learning methods	131
5.4	Experiment methodology	133
5.5	Results and evaluation.....	135
5.6	More data, better performance	140
5.7	Conclusion.....	142

CHAPTER 6 ‘SUSPECTRATE’ – A SPAM DETECTION SYSTEM 144

6.1	Introduction	145
6.2	Design goals	146
6.3	System flow and structure	147
6.4	Building and training models	151
6.5	Deploying and maintaining models.....	153

6.6	System data input	154
6.7	Feature extraction.....	155
6.8	Decision-making and output presentation.....	156
6.9	Implementation details	158
6.10	Conclusion.....	160
CHAPTER 7 CONCLUSION AND FUTURE WORK.....		161
7.1	Conclusion.....	162
7.2	Research limitation.....	164
7.3	Future work	165
APPENDIX A EXAMPLE OF A TWEET JSON DATA SAMPLE.....		167
APPENDIX B PYTHON LIBRARIES USED IN BUILDING THE SYSTEM.		171
APPENDIX C EXAMPLE OF A VISUALISED TREE MODEL		173
REFERENCES.....		174

LIST OF FIGURES

Figure 1.1 Thesis structure.....	29
Figure 2.1 Spamming units and stages [15].....	36
Figure 2.2 Search results for ‘buy Twitter accounts’ on Google search.....	40
Figure 2.3 Supervised machine learning general scheme [65] Figure consist two main parts training and prediction	46
Figure 2.4 Unsupervised machine learning general scheme [65] The coloured boxes represent the different clusters the dataset is split into	47
Figure 2.5 Model complexity against testing and training performance [69] Increasing complexity could enhance the prediction to a limit where the test sample error gets higher.....	48
Figure 2.6 SVM trained with samples from two classes green point are the support vectors that used to create the hyperplane to separate the two classes.....	51
Figure 2.7 K-NN example with k=5 As the circle represents unknown class data need to be predicted.	51
Figure 2.8 Random Forest generic mechanism. The whole dataset is divided into n samples and each sample is used for building a singular DT. Then in the final stage, each model prediction is combined for the final prediction.....	55
Figure 2.9 Boosting general mechanism the whole dataset used in all n iterations of the training of boosting models	56
Figure 3.1 Key phases of the research methodology. The collection phase is where tweets are imported into the database. This is followed by feature extraction and selection, and then building models and evaluating them.	72

Figure 3.2 General techniques used and their flow the flow represents the sequence of techniques used in processing incoming data till storing it in the database	75
Figure 3.3 Data collection and features extraction workflow crossed tweets represents tweet that got deleted by twitter	76
Figure 3.4 Labelling method used to build DS2 ground truth dataset, that involved extra manual validation	78
Figure 3.5 Labelling tool developed and used in labelling the dataset In the bottom right corner of the screen shot image there are three buttons: green (normal), red (spam) and yellow (unsure)	79
Figure 3.6 Cross-validation sampling method	82
Figure 3.7 Selenium web drivers web browsers logos represents the capability of selenium to use different software web drivers	87
Figure 3.8 Malicious/spam content hidden content behind multiple redirected URLs starting by un-blacklisted URL and ending with blacklisted one.....	90
Figure 3.9 Example of a popup window used for advertising	92
Figure 3.10 Example of ads abusive web page two column web page as both contains ad text and banners	94
Figure 3.11 Research methodology key phases	104
Figure 4.1 The effect of the number of trees parameter on the performance of the spam classification.....	113
Figure 4.2 The effect of the tree max depth parameter on the performance of the spam classification	115
Figure 4.3 The effect of the leaf size parameter on the performance of the spam classification	116

Figure 4.4 RF classification performance based on the selected features. The vertical axis is the performance in Recall and the horizontal axis shows number of features as it is decreasing from left to right. The yellow horizontal line is the model performance using all the features.....	125
Figure 4.5 RF incremental learning curve according to DS1 random forest performance according to the size of training data used	126
Figure 5.1 Comparison of DS1 and DS2 features importance based on two feature ranking methods	131
Figure 5.2 Models' performance according to F1 metric model on the left XGB is the highest model compared with varied combined models groups	137
Figure 5.3 Methods used for combining predictions of several machine learning models.....	138
Figure 5.4 Top models vs top combined model	140
Figure 5.5 Learning curves for ensemble learning models using the F1 metric. The figure shows the classifiers' performance behaviour as more data is used for training.....	141
Figure 6.1 System's main components flow starts from system web interface and ends by sending email to the client, small grey area represents the internal retraining process that conducted periodically.....	148
Figure 6.2 Feature extraction internal processes extracting feature process start by retrieving URL from the database and ends back with all extracted features to the database.	150
Figure 6.3 System's ability to add new models green box shows a new deep learning model added to the classifiers stack	153

Figure 6.4 Input file one column csv file, each row contains a tweet ID	154
Figure 6.5 Web form for uploading the dataset three fields required name, email and dataset file.....	155
Figure 6.6 Output file two columns csv file, first contains tweet id then suspicious rate	157
Figure 6.7 Web form to get the status of user order two fields required (email and serial number) to retrieve order status	157

LIST OF TABLES

Table 2.1 Examples of distance measurement metrics	52
Table 2.2 List of Twitter features adopted by previous studies [90]–[93].....	60
Table 2.3 List of lexical, host-based and page-content features [56][110].....	64
Table 2.4 Example of phishing URLs [112]	65
Table 3.1 Comparison between DS1 and DS2.....	80
Table 3.2 Sources of features used in building machine learning models.....	83
Table 3.3 Common features used in literature	84
Table 3.4 User information features	85
Table 3.5 Tweet features.....	86
Table 3.6 Redirections’ observation features and web page content.....	88
Table 3.7 WHOIS information features.....	95
Table 3.8 Confusion matrix	98
Table 4.1 Overall performance (average of ten experiments) using one classifier for all attributes	109
Table 4.2 Random forest main hyper-parameters.....	111
Table 4.3 Ranking of features based on information gain, Gini index and mean decrease average	123
Table 5.1 Common features and classifiers used in the literature (algorithm with highest performance identified by bold type).....	132
Table 5.2 Definition of metrics	135
Table 5.3 Results of models using stratified 10-fold cross-validation method.....	136

Table 5.4 Model sets used in combination methods.....	137
---	-----

Table 6.1 Models used in first version of SuspectRate system	152
--	-----

GLOSSARY OF TERMS

Term	Definition
OSN	Online social network, e.g. Twitter, Facebook and LinkedIn
Cybercrime	Crime that involves a computer and a network
Spam	Unwanted content typically sent to a large number of users for the purposes of advertising, phishing and spreading malware (Oxford Dictionary)
Dataset	All the data collected during the study, such as tweets and crawled web pages
ML	Machine learning
URL	Uniform resource locator
VirusTotal	Internet antivirus and internet security checker service
Domain name	Examples: microsoft.com and google.com
Web server	Where a website is hosted
RF	Random forest
XGBoost	eXtreme Gradient Boosting
Botnet	Network of connected software/devices used for illegal purposes

Spambot	Software hosted on a victim's computer or a controlling social network account that is owned and controlled by a spam master
IP address	Internet protocol address
GitHub	Source code storing and version control using Git
SVM	Support vector machine
Domain WHOIS Info	Information about a registered domain name, such as registrants, registrar, registration date, and expiry date
HTML	Hypertext markup language
Retweet	Twitter term meaning that a user shares another person's tweet to his followers
Followers (Twitter)	They see your tweets in their Home timeline whenever they log in to Twitter ¹
Following (Twitter)	Following someone on Twitter means subscribing to their tweets as a follower
K-NN	k-nearest neighbours algorithm
NB classifier	Naïve Bayes classifier
Sub-domains	A name that could be added to the original domain but separated with a full stop. For example, in scholar.google.co.uk, scholar is a sub-domain
CSS	Cascading Style Sheets

¹ <https://help.twitter.com/en/using-twitter/following-faqs>

Python	High-level programming language
Web crawler	Software with the purpose of opening URLs and reading and storing opened web pages
Selenium WebDriver API	Using programming language to automate a web browser (Chrome, Firefox, etc.) as if a normal user is using the browser
Web page	A document commonly written in one or several languages such as HTML, CSS and JavaScript. A web page is accessible through the internet or other networks using an internet browser
DS1	Dataset used in training and testing experiments in chapter 4
DS2	Dataset used in training and testing experiments in chapter 5
Twitter account suspension	Twitter internal detection system that aims to delete accounts that are not following Twitter's rules. Suspension can be due to spam activities, hate speech, etc.
Ad blocker	Web browser extension that works on filtering and denying contents based on predefined lists that mainly block advertisements in the web pages

Twitter protected account	Twitter account where their owners do not want to share their tweets and activities with the public, so only approved followers will be able to see the tweets
Twitter verified account	A blue badge attached to public interest people account for authentication ²
ReliefF	Feature ranking algorithm
LightGBM	Open source gradient boosting tree algorithms built by Microsoft
CatBoost	Open source gradient boosting tree algorithms built by Yandex Technologies
Popup window	Window that opens alongside web pages with the loading, unloading or any other actions
MongoDB	An open source NoSQL database management system
Tweet ID	A unique number that every tweet in Twitter has to have as an identification

² <https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts>

ACKNOWLEDGEMENTS

I would like to thank my country Iraq and my scholarship provider, the Ministry of Higher Education and Scientific Research, Republic of Iraq, for offering me the opportunity to undertake this study.

I would like to express my sincere gratitude to my advisor Prof. Peter Andras for his continuous support during my PhD study and for his patience, motivation and immense knowledge. His guidance helped me throughout the research and the writing of this thesis.

I would also like to thank my co-supervisor Dr Ed de Quincey for his constant support and availability and his constructive suggestions. Special thanks to Mr David Collin for his support, especially during the first year of this research.

I would like to extend my thanks to all the staff and students at Keele University and the School of Computer Science for their advice along the way. Special thanks to Keele IT Services for providing me with a virtual private server.

My deepest gratitude goes to my beloved parents Fadhil and Iman. I would have achieved nothing without their love, support and encouragement. Special thanks to my brother Safaa and my beloved nephews Mustafa and Muwafaq.

To my wife Huda, who will always be a source of support and love, I thank you for being patient and understanding. The most important contribution to my PhD was my daughter Jana, who joined us for the first year of this journey. Thank for your smiles and joy. You will always be my little princess.

All praise and gratitude are due to Almighty God for everything.

Mohammed Al-Janabi

2018, Newcastle-under-Lyme, UK

ABSTRACT

This thesis proposes the use of several supervised machine learning classification models that were built to detect the distribution of malicious content in OSNs. The main focus was on ensemble learning algorithms such as Random Forest, gradient boosting trees, extra trees, and XGBoost. Features were used to identify social network posts that contain malicious URLs derived from several sources, such as domain WHOIS record, web page content, URL lexical and redirection data, and Twitter metadata.

The thesis describes a systematic analysis of the hyper-parameters of tree-based models. The impact of key parameters, such as the number of trees, depth of trees and minimum size of leaf nodes on classification performance, was assessed. The results show that controlling the complexity of Random Forest classifiers applied to social media spam is essential to avoid overfitting and optimise performance. The model complexity could be reduced by removing uninformative features, as the complexity they add to the model is greater than the advantages they give to the model to make decisions.

Moreover, model-combining methods were tested, which are the voting and stacking methods. Both show advantages and disadvantages; however, in general, they appear to provide a statistically significant improvement in comparison to the highest singular model. The critical benefit of applying the stacking method to automate the model selection process is that it is effective in giving more weight to more top-performing models and less affected by weak ones.

Finally, 'SuspectRate', an online malicious URL detection system, was built to offer a service to give a suspicious probability of tweets with attached URLs. A key feature of this system is that it can dynamically retrain and expand current models.

CHAPTER 1

Introduction

1.1 Introduction

Online social networks (OSNs) have become one of the main mediums of communication. Statistics show that in the first quarter of 2018, Twitter had around 336 million active users monthly generating more than 500 million tweets per day [1]. This number made it a fertile environment for the dissemination of malicious content and illegal monetary gain by spammers [2]. Social network administrators are responsible for protecting OSN users from being exposed to malicious or spam content in networks. In total, 60 per cent of social network users have received or been exposed to spam content [3]. Spam is the generic term for all types of unsolicited and harmful content, which includes phishing links, malware or links to illegal or fake products [4], that is typically sent to a large number of users. Some additional social network activities could also be described as spamming, such as gaining fake popularity and following or liking a large number of users or content to gain attention.

The ability to distribute content in real time to thousands of recipients has made it difficult to resolve the issue of spreading unwanted content and suspicious links, especially that OSNs dealing with big data [5]. Specifically, the ease of creating accounts on OSN sites and the simplicity of the distribution process make them a perfect environment for the dissemination of spam content. For example, studies show that the proportion of spam in all Twitter content has reached an all-time high that stands at approximately 10 per cent of all content produced [6]. Coupled with the rapidly increasing number of active OSN users, this has emphasised the need for secure OSN platforms whereby operators frequently monitor any ‘abnormal’ activities in their network to detect attacks [7]. However, cybercriminals search continually for attack opportunities, which rely on previously undiscovered vulnerabilities, for which there is

no effective defence. A newly deployed attack technique is called a zero-day attack on the day that it is first applied [8]. Furthermore, the complexity of such activities is exacerbated by the social nature of OSNs, whose users tend to trust content that has originated or emerged from their peers' content [9]. This adds further difficulties to the missions of security administrators and detection systems.

At the heart of the problem is the emergence of black markets, which increasingly facilitate the spreading of illegal services, such as the deployment of spam campaigns and the selling/buying of fake accounts, compromised accounts or even infected computers. An increasingly important way in which cybercriminals engage in illegal activities is through the simple creation of fake OSN accounts [10]. There is a constant war between cybercriminals and security experts, as every time OSN operators detect and block a spamming attack, cybercriminals find a new way to leverage social networks and abuse their services. There is a gap of time which cybercriminals can take advantage of before the OSNs are able to detect new attacks. In particular, due to the real-time nature of social networks, this makes it more difficult to monitor and protect users and content.

1.2 Background

The use of OSNs now exceeds that of email in the propagation of harmful communications (e.g. fake news, advertising for illegal services, unwanted adverts) [11]. The same study shows that between 2013 and 2014, spam content in social networks increased by 658 per cent. Studies indicate that the main vehicle used to propagate malicious attacks in emails is the same attack botnets used by more conventional email attacks [11], [12]. A botnet is a group of compromised computers/accounts that are under the control of a cybercriminal [13].

The ease with which OSNs can be used for malicious activities and the relatively long time needed to detect/suspend them [14] make a compelling business case for such criminals. The consequent profitability has led to the expansion of a new industry catering to the needs of cybercriminals who target OSNs [15]. To protect users' personal data and privacy, many issues should be considered to address several key challenges in OSNs, including:

- Preventing spammers from creating accounts and/or suspending them in the shortest possible time
- Detecting spam campaigns and stopping them before they reach a bigger audience
- Detecting abuse such as spreading irrelevant content to trending hashtags or accounts
- Detecting fake followers which are gained by using illegal methods.

1.3 Motivations

Many studies have attempted to analyse and tackle the high and rapidly increasing spam content in recent years [16]–[18]. Although researchers are seeking to mitigate it, to date, no work has succeeded in eliminating it. According to Nexgate's statistics, OSNs' spam content increased by six times since mid-2013 [11]. These problems (described in section 1.1) have been the principal motivation for this research, which seeks to reduce or even eliminate spam content in OSNs.

1.4 Problem statement

Based on the discussion in the preceding sections, this research classifies spam from two perspectives: OSN user and operator. Accordingly, this has led to several problems, which are summarised below:

- (i) **User's perspective** – this refers to unwanted content that is distributed by fake and compromised accounts for commercial reasons or malicious activities. Such content could have Uniform Resource Locator (URLs) that link to harmful software or phishing sites.
- (ii) **Operator's perspective** – spam increases the load on an OSN platform and creates a significant drain on resources in addition to the loss of trust by its users.

Consequently, this research proposes to resolve these problems by detecting spam content that emerged from both fake and compromised accounts.

1.5 Objectives

The main aim of this research is to develop a computational framework to reduce the spread of spam content by proposing a practical approach to identifying spam tweets in Twitter based on the selected features and characteristics. The main objectives of this research are:

- To identify, study and analyse methods and features that enable the identification of spam content on OSN platforms. An important part of this work entails the review of the impact of key parameters on the learning process, with particular attention to avoiding overfitting.
- To develop efficient and scalable OSN spam detection framework with predictive data analytics which are capable of summarising and describing

patterns in the collected data. In particular, the objective is to identify data analysis methods that make possible the development of high performance, dynamically adapting spam identification and filtering.

1.6 Research questions

1. What combination of features enables improved detection of spam on social networks compared to current approaches?
2. How can the performance of combinations of classification methods such as random forest and gradient boosting trees classification methods be improved for spam detection in the context of social networks?
3. How can a system be designed using an optimised set of features and an optimised combination of classification algorithms to deliver high usability combined with improved spam detection in the context of social networks?

1.7 Main contributions of thesis

The primary aim of the thesis was to develop a spam detection system based on machine learning models that are adaptable to future spamming tricks and activities. The points listed below are the main contributions of this research:

1. It built two ground truth datasets using different labelling standards that would help researchers' in training models. These datasets are downloadable on GitHub via the following URL: <https://github.com/mohfadhil/suspectrate-datasets>. Both datasets were collected through Twitter real-time stream tweets; however, each used a different labelling mechanism. Dataset 1 assumed that all tweets that are deleted are spam tweets with no manual process. Dataset 2 used the previous labelling mechanism but with extra manual validation.

Furthermore, more features were introduced and features were discarded from Dataset 1 due to the pilot study conducted on the effectiveness of the features. (Chapter 3)

2. It developed the first labelling method, which is introduced in chapter 3, where three validation processes used Twitter suspension, VirusTotal blacklists and manual labelling, as this method showed more accurate labelling standards. (Chapter 3)
3. It conducted a systematic analysis of the most important parameters in random forest that could make a difference in the model's performance and overfitting/underfitting status. (Chapter 4)
4. It applied new algorithms that have been studied and shown good performance compared to traditional algorithms to the datasets. Moreover, it gave details about the algorithms' hyper-parameters assigned to support the research reproducibility and open science. (Chapter 4)
5. It used stacking and voting methods to automate the process of model selection, so a human decision will not be needed to choose the best model. (Chapter 5)
6. The proposed novel set of features were derived from web page content and behaviour. The novelty is that the author consider content that appears at the landing web page and while reaching the landing page. (Chapter 3)
7. The system was built from scratch using Python and many other 100 per cent open source libraries. The developed framework applied the main machine learning system components by collecting data and extracting and selecting features. Then the problem of model selection and evaluation was handled by using extra meta classifier. SuspectRate was able to retrieve tweets with URLs

and store them in a database and then perform analysis and use it against a pretrained model or combined models. The system is open source at the researcher GitHub repository: <https://github.com/mohfadhil/suspectrate>. (Chapter 6)

8. The research experiments have been published in two peer-reviewed conference papers:
 - Al-Janabi, M., Quincey, E., & De Andras, P. (2017). Using supervised machine learning algorithms to detect suspicious URLs in online social networks. Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017 (pp. 1104–1111).
 - A systematic analysis of random forest based social media spam classification, Mohammed Al-Janabi, P Andras, International Conference on Network and System Security, 427-438.
9. Several talks have been/will be given regarding this research:
 - Four talks at postgraduate research (PGR) days from 2014 to 2018
 - Three talks at the Keele symposium of postgraduate studies in 2016, 2017 and 2018.
 - Posters presented at the Keele symposium of postgraduate studies in 2016, 2017 and 2018.

1.8 Structure of thesis

In chapter 1, the research questions and aims are stated. Chapter 2 outlines in detail the situation of current spam and anti-spam detection mechanisms and the author's role to reduce the spam percentage in OSNs. Furthermore, all the machine learning models

mentioned in this thesis are described in chapter 2. Several related studies have been reviewed and categorised the models used in building their machine learning models based on their selected features.

Chapter 3 describes the road map of the main experiments that the researcher conducted during this study, how data was collected, and the feature extraction and labelling process. It also describes how the model selection process was conducted. In this chapter, the technical process and tools used in the feature extraction and storing the dataset are also outlined.

Chapter 4 describes the first pilot study conducted using several machine learning algorithms to gain a better understanding of which algorithms would better suit the research classification problem and dataset. It also describes the model enhancement procedures conducted, such as model hyper-parameter tuning and feature selection.

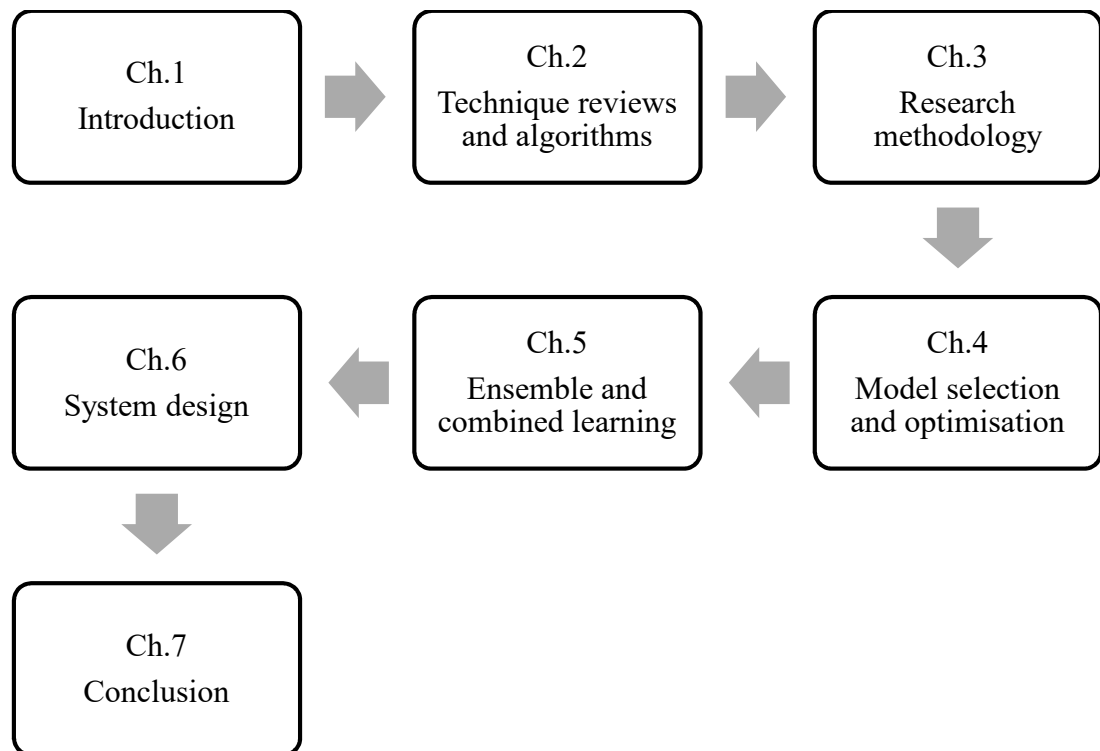


Figure 1.1 Thesis structure

Chapter 5 focuses more on ensemble learning algorithms and compares the top algorithms in the field. It describes two experiments on combining boosting and bagging trees in one stacking and bagging model.

Chapter 6 shows the developed system to rate suspicious URLs to help users, researchers and companies to label their data or at least clean it.

The discussion and the limitations of the thesis are presented in chapter 7. The main conclusion of the thesis and suggestions for future work are also presented in chapter 7. The future work offers recommendations for new research studies that have similar research aims.

CHAPTER 2

Literature Review

2.1 Technical background

Generally, the mechanism of posting content and how it reaches other users on the network is quite similar in most OSNs, i.e. following users aggregate the content they produce into a personalised news feed. Furthermore, OSNs have search capabilities that enable a user to search for certain keywords in the content of followed accounts or public content. In addition to the search feature, there is an approach that has been used in OSNs to make following a certain thread or topic easier, which is called a hashtag. A hashtag is non-separated characters that refer to a certain topic or trending event. It starts with the symbol ‘#’, which is followed by a string of characters. Hashtags were first used on Twitter [19]; however, this technique is now used in most user-generated data websites [20]. In this thesis, Twitter was used as the OSN case study; however, the concept of this work could be used in any other OSN if it has a similar structure for distributing content. The common reason for using the Twitter platform as a data source is that Twitter’s data mostly has public access. However, there are accounts called protected accounts³; their data and tweets are protected and cannot be reached by collection software. These protected accounts make up less than 6 per cent of all accounts, which means that almost 94 per cent of all Twitter data is public [21]. Twitter supports researchers by providing software tools to facilitate the data collection process. Moreover, the openness of Twitter and the huge amount of data that can be accessed

³ Public and protected tweets <https://help.twitter.com/en/safety-and-security/public-and-protected-tweets> [accessed April 2018]

are vital features that motivate most researchers in this field to use Twitter as a data source.

In this chapter, there will be a review of general threats that OSN users might be exposed to and the current mitigating techniques used, such as blacklists and machine learning-based detection models, to provide a clear understanding of the advantages and disadvantages of each detection technique. Moreover, there will be a review of several relevant studies and techniques/methods used in building spam/malicious URL detection systems.

2.2 What is spam content?

In general, spam is defined as the unwanted content that is sent to a large number of users. This definition is used for spam in emails; however, the main concept of spam is the same in any other messaging or content-sharing platform. In the context of OSNs, spam could be any unwanted content or even duplicated content. Spam content can be in many forms, such as illegal product advertisements, false news, phishing or URLs that lead to drive-by download attacks [22], [23]. To understand why these types of content are considered as spam in OSNs, the following types of content are considered.

Phishing sites: phishing sites are cloned websites of real popular websites such as facebook.com, twitter.com or even the popular banks. Online scammers spread content with URLs attached over OSNs to thousands and sometimes millions of accounts in OSNs to lure the account holders to visit these cloned sites. Many users might notice that there are some suspicious characteristics in these phishing sites, for example, the URL path contains sub-domains such as ‘twitter.com.twt.com/login.php’. Unfortunately, however, a percentage of users will not notice and log their username/password, which goes directly to the spammers/hackers. This attack is one of

the most popular attacks of spammers because of the simplicity of creating cloned sites to obtain users' account usernames and passwords.

Fake software: like phishing websites, fake software or trojan horse software is software that falsely claims that it will give users services such as boosting the speed of their devices or increasing the RAM or hard disk space. There are even cases where the fake software pretends that it has found some viruses on the user's devices and cleans the devices. This type of software is usually promoted on sites with a very low reputation that spammers created and advertised using their fake/compromised account to OSN users.

Fake news: fake news is when someone posts on OSNs or publishes on his website partly falsely edited facts or total fake news [24]. This was started by people who lure users to visit their site. They post exaggerated titles and could even attach posts with manipulated photos that attract the user to click on attached URLs. It started as a way to gain more internet traffic to obtain increased website ad earnings; however, it has been used for other reasons as well as to gain profits. Currently, there are investigations relating to whether this unlawful way of spreading news has been used to target a large population on very specific events, such as a presidential election (e.g. the United States presidential election, 2016) [25], [26].

Clickjacking web pages: Clickjacking is one of the new browser attacks that online scammers use to hijack users' clicks. A scammer can build web pages that have clickbait pictures or links that lure users to click on them, but in reality, they are clicking on some other JavaScript trigger event [27]. This technique is used for 'likejacking', which is one of the illegal ways that scammers increase the number of followers of a certain account/Facebook page. There is a black market paid service for increasing

Facebook page likes or Twitter followers, so they take users' clicks and use them as if the user is clicking to like a page script code or following an account on Twitter.

Illegal advertising: although most OSNs provide a legal channel for promoting content on their network, spammers illegally flood the network with advertisements. The reason for this is either that they want a cheaper method of advertising or that the content they want to promote is not accepted (i.e. uncertified medical equipment or medicine).

Pornography, dating and adult content sites in general are among the sites that have a high percentage of spam content distributed over social networks. Social networks have different rules on this type of content; however, in general, there is a restriction on spreading this type of content in OSNs. Pornographic pictures are usually used as click-bait to lure users to click the URL to get more content or watch a full porn video. Moreover, pornographic content could be hidden as it needs a Flash Player to view it in the user's browser, so the user is redirected to a phishing website to download a fake Flash Player.

2.2.1 Twitter's spam rules

None of the spam types mentioned in the previous section are accepted to be posted on Twitter. Furthermore, Twitter monitors users' behaviour to detect any abusive activities in the network, which is mentioned in Twitter rules page⁴. Consequently, an account could be suspended due to its content or its actions in the networks [28].

⁴ The Twitter Rules, <https://support.twitter.com/articles/18311> [accessed May 2018]

According to Twitter's suspension rules web page, there are three major reasons to suspend a user's⁵ account or delete a tweet:

1. breaking copyrights
2. abusive tweeting activity
3. spreading malicious and harmful content.

Twitter has also restricted the spreading of sexual/nude pictures over the network and tried to hide them and give sensitive media warnings to users. Moreover, Twitter does not allow users to use nude pictures as profile or cover images, and users are not allowed to send messages containing pornographic content to people. If nude pictures or sexual text are used as part of click-bait strategies by a spammer, Twitter tries to stop this.

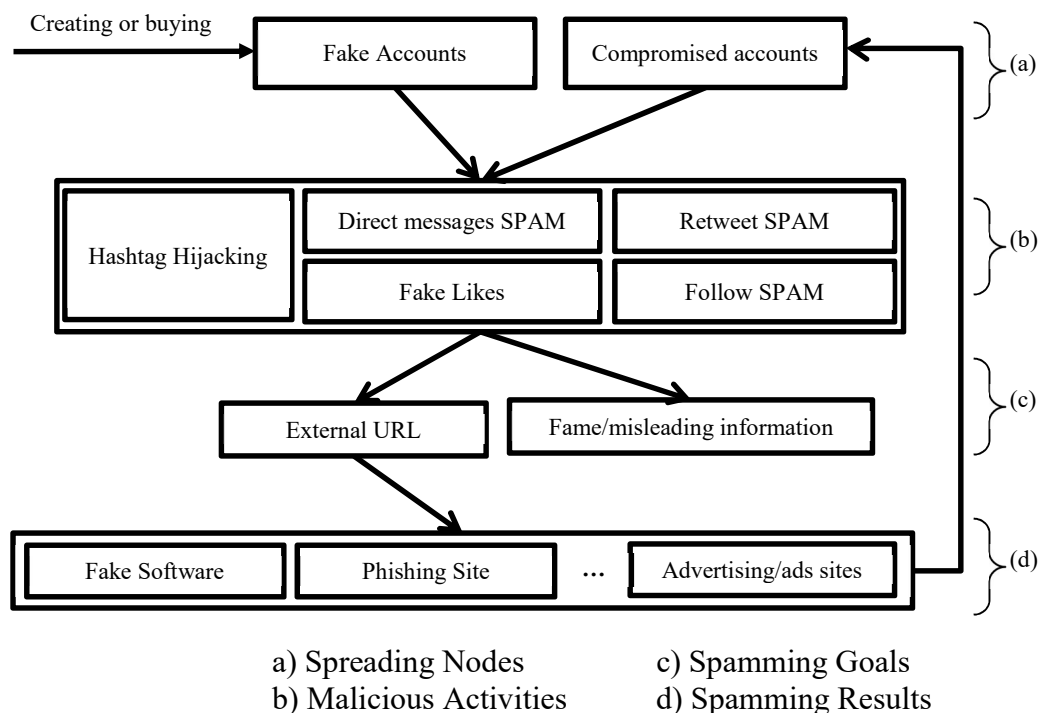


Figure 2.1 Spamming units and stages [15]

⁵ There is also the chance that the user deletes the tweet.

2.3 Spam mechanism

To understand these activities and the measures used to counter them, a definition of the key terms used in relation to social networks needs to be provided and the most common practices need to be explained. An explanation of how fake accounts are created and how accounts are compromised will be provided. Figure 2.1 [15] illustrates how fake accounts play a key role in spam content distribution and describes the main spamming elements.

Spreading Nodes: Every account in an OSN is considered as a node, and every node has a relationship with others [29]. The more nodes spammers have, the more spam content they can spread [30]. Therefore, spammers are one of the major causes of the increase in fake/compromised accounts in OSNs. Due to detection systems that OSNs have developed to prevent spam content spreading, many associated accounts are suspended every day [31]. The illicit industry of creating and selling accounts is still active to recover suspended accounts and help spammers to have enough active accounts for their spam campaigns [32]. As shown in Figure 2.1, compromised accounts also play a role in spreading spam content although they have originated in a different way. Compromised or infected accounts are legitimate; they are created by normal users, but somehow spammers have the ability to control them. In the spamming industry, infected accounts are more valuable than fake ones [32], as it is more difficult for OSNs' detection systems to detect and suspend compromised accounts compared to fake ones. Therefore, spammers tend to focus their effort on infecting legitimate accounts with the aim of increasing the number of compromised accounts under their control.

Spammers often also use fake accounts, which are cheap to buy in online black markets, to conduct their spam campaigns. Several studies show that about 10 per cent of all OSN accounts are fake accounts [33]. Cybercriminals control these fake accounts using computer programs to perform automated operations using bots, which act as legitimate users. Bots employed in this way have become known as social bots. They are essentially programs which simulate the activity of a typical user on a social network [4], [7], [34]. For example, they are able to post, message, vote and share [35].

Malicious Activities: Spammers have various techniques and tricks to increase their audience in OSNs; one of the most commonly used techniques is known as hashtag hijacking [35]. Spammers exploit trending topics by posting/tweeting using these trending keywords or hashtags, giving them a wider audience who follow those trending topics [36]. Furthermore, many malicious activities could be conducted by spammers to lure users to click on their malicious URLs.

Spamming Goals: Deliver spam content to the targeted users is the primary task, which is done by redirecting the user to a suspicious source outside the OSN site. The URL usually used had been shortened once or several times. This link can refer to a phishing page, scam or drive-by download attacks. Recently, a study has shown that the high number of URLs that are spread by a Twitter account can often be under the control of a spambot [35],[37]. The high percentage of spam tweets that contain external links or URLs gives an indication that spreading URLs is a major task for spammers. Moreover, some OSN activities that can come under the classification of spam are fake likes, fake followers, and spam retweets. Some spammers also use spamming services to get more attention or to increase their followers number (fake fame) on OSNs or spread misinformation [38].

Spamming Results: The primary objective of spam campaigns is not only to let users see spam, but also to get them to click on the attached links. Encouraging users to click on those URLs requires several tricks by spammers to deceive them by luring them with pornography, celebrity scandals, free software, discounts codes or bargains deals [6]. These links may point to web pages that lead to drive-by download malware attacks to steal users' information using fraud or phishing sites [37][2][39].

There are several method that spammers can use to create new URLs with no historical profile, such as:

- **URL-shortening services:** these are web services that after submitting a URL to the services, provide a new short URL that points to the same original URL [20]. These short URLs are mainly used in social networks with a limited content length such as Twitter. Currently, many shortening services, for example, bit.ly and tinyurl.com, are commonly used in OSNs [40].
- **Cheap domain and hosting services [41]:** creating new websites requires two main elements, which are a domain name and online space to host websites files. Domain names are cheap nowadays, and spammers can buy a domain name for less than £10 [42], [43].

The above services are responsible for a high percentage of the spam content distributed over social networks. Although Twitter uses blacklists, which are suitable for real-time detection, unwanted content still finds its way into the network [44].

2.3.1 Sybil attacks and fake accounts

Spam industry based on the number of accounts controlled by spammers, these account as stated could be either compromised or fake accounts [45]. The method that

attackers use to create fake accounts in OSNs using fake identities is called a Sybil attack. This type of attack is very common in OSNs, in which a single user can have thousands of fake accounts so they gain higher visibility by spreading more content in the network [46]. According to a previous study that focused on the Sybil accounts in Twitter, it was found that out of the total accounts monitored, around 2 million or 9 per cent get suspended as they are considered to be Sybil accounts [47]. This is close to the 10 per cent that Twitter officially announced as the spam percentage in the content [48].

2.3.2 Fake accounts and the black market

As discussed in the previous section, in general, the spamming industry relies entirely on the nodes (accounts) used to spread spammers' content. As OSNs suspend accounts that are involved in spamming activities, the spamming industry needs to generate enough accounts for their spam campaign. Creating accounts and offering them for sale in the black market has reinforced the spam industry. Thomas [15] conducted a study on the impact of the black market and how it facilitated the process of spreading spam content using fake accounts.

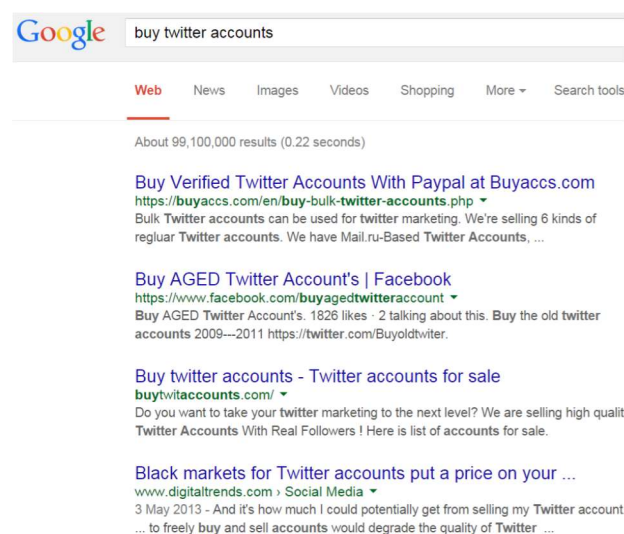


Figure 2.2 Search results for 'buy Twitter accounts' on Google search

The researcher studied the value of fake Twitter accounts and found that its market worth was between US\$0.01 and US\$0.20 [15] for one account. He also reported that it has become increasingly easy to purchase fake accounts in bulk (generally thousands) online. The continuity and availability of fake accounts have contributed to spamming activities in social networks. Figure 2.2 is a screenshot of Google search results using the term ‘buy Twitter accounts’ in April 2015, showing many web sites that promise to sell verified twitter fake accounts as a service.

2.3.3 What are spambots?

What makes the problem of the high percentage of Sybil/fake accounts in OSNs more complicated is the smart programs (bots) that control those fake accounts. In general, bots are computer programs that can automate actions and responses based on certain rules prespecified by the person who controls the bots, who is referred to as a ‘botmaster’. Bots are assigned to control the thousands (sometimes millions) of fake accounts that belong to the spammer who is the botmaster of those spambots. Bots are also used for different attacks such as botnet attacks by controlling infecting machines and deploying attacks such as distributed denial of service (DDoS) and or use them as email servers for spamming.

A spambot in Twitter is a computer program that is used to perform in a similar way to a normal OSN user to perform normal activities such as tweet, retweet, favourite, and follow/unfollow accounts. Automating these actions has helped many spammers to control their large number of fake accounts. Moreover, the near to normal behaviour that the current smart bots use to mimic normal users is making it difficult for them to get caught by the Twitter suspension system. Later in this chapter, some evading tricks

that spambots use to make it less likely that they will be detected by current detection methods will be discussed.

2.4 Countermeasures

Historically, countermeasures have gone through a similar evolution to that which has occurred in attempting to deal with email spammers. As the attacks become more sophisticated, new countermeasures need to be adopted to address such attacks. Two general methods are used to reduce or stop malicious content spreading through email and OSNs, which are knowledge engineering and machine learning [49]. Knowledge engineering is described as a group of predefined rules and limits for allowing and denying content [50]. The blacklist technique is a common example of knowledge engineering methods that are used in OSNs. OSNs use blacklists to accept or discard any content that does not follow the rules. These rules need to be regularly updated to cover all the new evading techniques that spammers use. Unlike blacklists, machine learning does not need pre-set rules, as it learns implicitly from the thousands of input samples. The next section will discuss in detail the blacklist and machine learning techniques used in OSNs.

2.4.1 Blacklist

The blacklist-based technique is widely applied in Web 2.0 sites, that is, sites used to facilitate the generation and sharing of users' content [15]. The spam filtering process requires several methods and layers, and the real-time blacklist technique is used as a first line of defence [51]. In terms of websites and social networks, a blacklist is a simple access control rule that allows user-generated content to be published if it does not contain any blacklisted keywords, text, images or URL [52]. Many products

have been used to detect malicious pages, such as Google Safe Browsing⁶, AVG Linkscanner, McAfee SiteAdvisor⁷, PhishingTank, URIBL, SURBL, and Web of Trust⁸ (WOT). This technique is based on users' content and account properties, which could be IP addresses, emails, and domain names [53].

The Google Safe Browsing API has more than 600 million users [54]. It is used every day, directly or indirectly, in browsers such as Google Chrome, Firefox, and Safari. Google Safe Browsing is a public blacklist database which has an API that facilitates the process of looking up and verifying URLs. The Google Safe Browsing blacklist is constantly updated with newly detected malicious sites, phishing, and malware pages. If a URL matches an entity in the blacklist, it will give an early warning and prevent the user from clicking on or accessing the site [55]. In addition to Google, several security companies produce blacklist detector products, such as AVG LinkScanner and McAfee SiteAdvisor. Both are free tools provided to protect users based on a blacklist technique. However, WOT depends on the crowd-sourced reputation gathered by internet users' experience in websites. All the services mentioned are capable of detecting URLs that are already known for malicious activities. This technique offers real-time detection with a low false positive rate. Despite the features mentioned, blacklist techniques cannot detect malicious content that has never been detected before. The blacklist databases vary in terms of source of updates, it could be accept users feedback (i.e web of trust) or rely only on their maintainer security research centres.

⁶ Safe Browsing by Google, <https://safebrowsing.google.com> [accessed May 2018]

⁷ McAfee WebAdvisor, <https://www.siteadvisor.com> [accessed May 2018]

⁸ Web of Trust, <https://www.mywot.com/> [accessed May 2018]

Currently, OSNs use blacklist detection systems to check users' content, such as posts, tweets and links, before publishing any content [56][12]. Conceptually, blacklists are a simple method applicable to all web services that accept users' content. The blacklist real-time detection feature is vital in the age of high-speed data exchange. However, the false negative rate due to the zero-hour attacks is the main shortcoming of this technique [57]. Blacklist solutions consider any URL that is not blacklisted as benign, but this decision might not be valid permanently. Due to the real-time feature of social networks, spam campaigns achieve 90 per cent of their goal within the first 48 hours [12]. The main shortcoming of the blacklist technique is the time gap between detecting the 'unblacklisted' spam content, or the zero-hour threat, and the publishing time.

Due to the critical issue mentioned above, there is a significant need to upgrade this list on a regular basis, daily or hourly [15]. Upgrading the list is the responsibility of researchers or cyber security companies depending on who maintains these blacklist solutions. Blacklists used in all OSNs are useful to remove the content that has already been discovered and listed; however, the real challenge is when the platform's detection system receives a new unlisted URL/domain with no history. URLs/domains with no history get past the blacklists' filter and are distributed in real time to thousands of users. The time gap between content with suspicious content attached being distributed through the network and listing it on a blacklist is what security centres and researchers are trying to narrow [58].

2.4.2 Introduction to machine learning methods

The volume and complexity of data exchanged in OSNs and their real-time nature [59] make automated data analysis essential. Therefore, the need has arisen to

apply methods to enable machines to monitor and detect malicious content without any human intervention. The process of making a machine learn how to make decisions on its own is called ‘machine learning’. Machine learning refers to a group of methods focused on designing systems that can learn, predict and make decisions based on the input data [60], for example, discovering patterns in given data or acquiring training models by analysing sample data and then using these models to classify or predict subsequent data [61]. Machine learning methods are divided into supervised, unsupervised and semi-supervised learning [62]. In supervised learning, a training stage is required using predefined (labelled) data for the training stage [63]. Unsupervised learning algorithms are different in that they do not require training with pre-labelled data. These unsupervised learning algorithms attempt to extract insights and patterns often from huge amounts of unlabelled datasets and then cluster datasets into sub-groups. Semi-supervised learning training datasets, however, contain less labelled data and larger amounts of unlabelled data, so the model uses the small labelled data to train a model and then label the rest of the dataset. Each method has its own advantages and disadvantages, so choosing a method need to be after a comprehensive study of the problem and availability of labelled dataset.

Due to the lack of labelled training data, the labelling stage is usually performed manually by the authors [36]. For example, one would [64][36] manually label a portion of the collected data as either normal or spam.

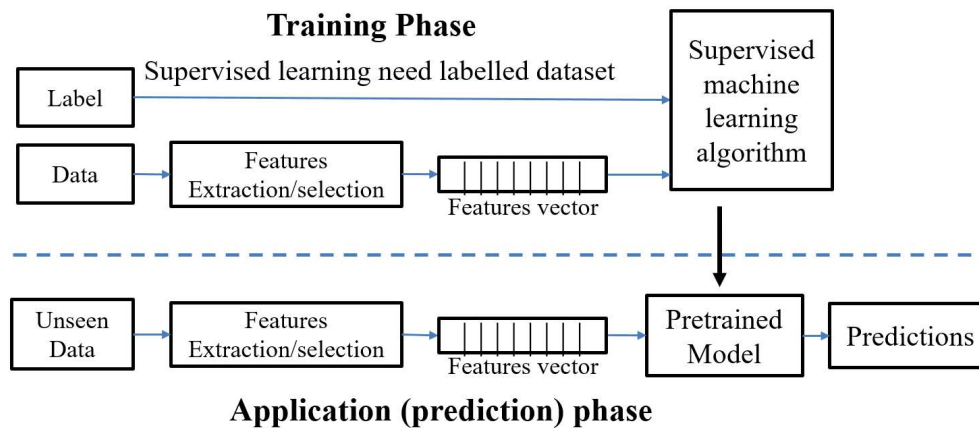


Figure 2.3 Supervised machine learning general scheme⁹ [65]
Figure consist two main parts training and prediction

In this section, several machine learning algorithms are investigated, Figure 2.3 and Figure 2.4 show the two main types, supervised and unsupervised, the main difference being that the first type required a pre-labelled dataset to train on to build the model. Pre-labelled data means an acceptable amount of data is predefined into categories (discrete numbers i.e. spam/non-spam and handwritten digit recognition) if the goal is classification or have a target of continuous number (i.e. salary and age) if regression. Training is conducted by finding the best formula from the features and target labelled so that a machine learning model can generalise rules learned from the training phase and apply them to an unseen dataset. The unseen data is data that has never been involved in any of the stages of building and training the model. However, in the unsupervised machine learning method, no labelled dataset required as model built based on how a dataset can be sub grouped into two or more groups each have commonality based on giving features.

⁹ <http://www.nltk.org/book/ch06.html>

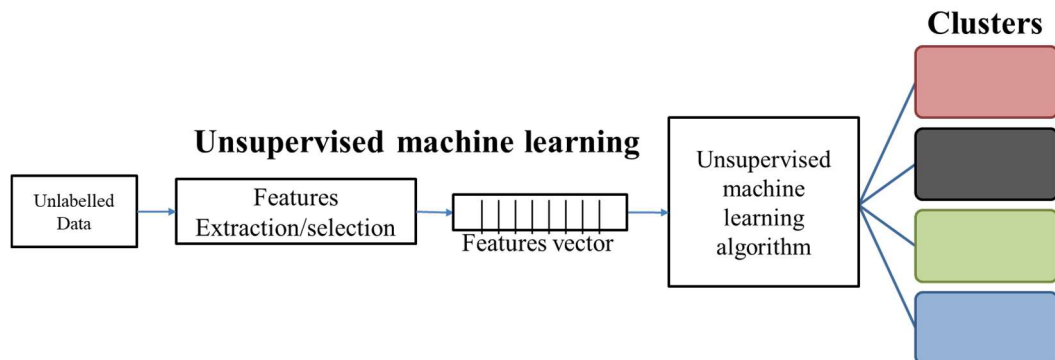


Figure 2.4 Unsupervised machine learning general scheme¹⁰ [65]
The coloured boxes represent the different clusters the dataset is split into

Both methods require pre-processing stage on inserted data whether data is labelled or unlabelled. Generally, one of the important aspects of machine learning is feature extraction and selection, which is the process of identifying sensitive characteristics from the input data [66]. So, after all before data inserted into a machine learning model it needed to be transformed into vector of numbers, which is the only format of data that machine learning can handle. Features selection is important stage in building any machine learning models but some models have advantages of handling features selection as a part of the internal model processing. For example, deploying deep learning models would not require feature selection in the pre-processing stage, as it performs the process of extracting and selecting features internally.

The features in the context of machine learning are prominent characteristics that contribute later in the discrimination phase. For this, choosing the best set of features available is an essential process. Selecting the most efficient set of features can

¹⁰ <http://www.nltk.org/book/ch06.html>

have significant impact on the performance of the machine learning algorithm [67]. The common feature selection methods identified in the literature include information gain, the chi-squared, and the F-Score [68][67].

Building a model with less important features has an impact on the model's simplicity and the time required for training and building. In general, the higher the number of features available, the higher the chance that the model can fall into the problem of 'overfitting', which often occurs as a result of the complexity of the model, when the model tries to customise to all the features and data cases in the dataset [66]. Therefore, the model performance will have very low bias and high variance. Although training shows the best performance, in the testing validation, it shows very poor performance.

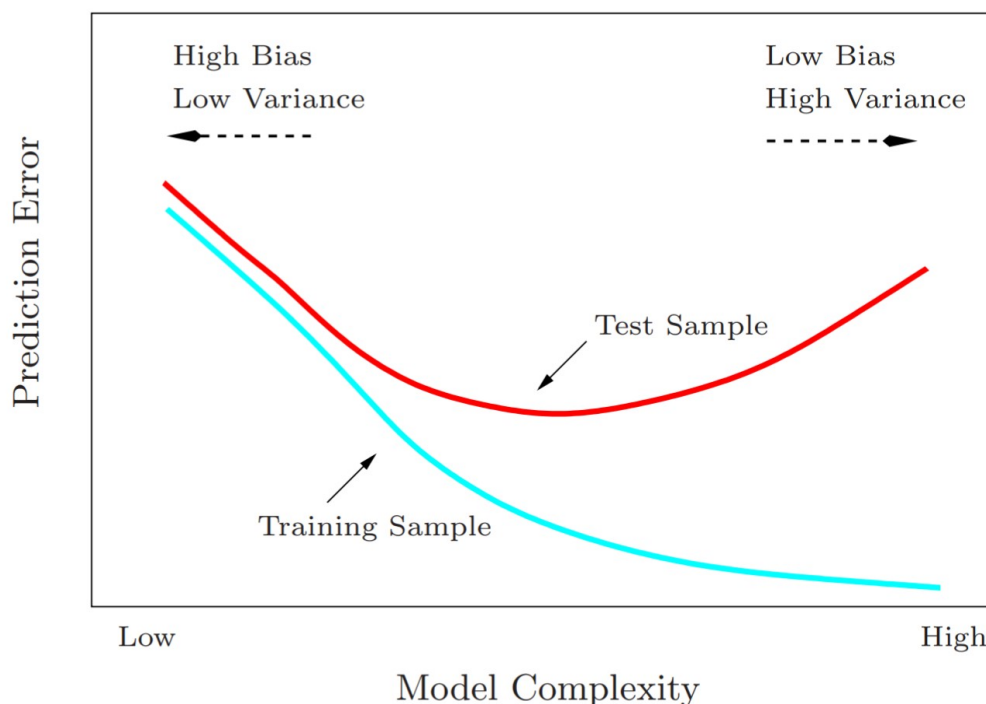


Figure 2.5 Model complexity against testing and training performance [69]
Increasing complexity could enhance the prediction to a limit where the test sample error gets higher

Underfitting is when the model does not get the maximum discrimination power from features as the model is too simple compared to the dataset. As this model has low variance but high bias, this means that the model performance during training and testing is quite similar and it does not perform well.

2.4.3 Machine learning algorithms

Machine learning algorithms are applied for spam classification and detection. Learning algorithms have proven [70] their ability to enhance the accuracy of spam detection in emails and OSNs. Selecting a suitable learning algorithm for a specific dataset requires study, as algorithms behave differently for different datasets. Much of the recent literature has used supervised machine learning algorithms, which, as will be mentioned later, require a preliminary training stage. Based on the review of the literature, several supervised learning algorithms used in a spam detection context are Naïve Bayes (NB), k-Nearest Neighbours (k-NN), Support Vector Machine (SVM) and Decision Tree (DT) classifications.

The NB classifier is a probabilistic model based on the Bayes rule. ‘Naïve’ refers to the assumption of conditional independence among given vector features $X = \{x_1, x_2, \dots, x_n\}$.

$$P(C_i|X) = \frac{P(C_i) * P(X|C_i)}{P(X)} \quad (1)$$

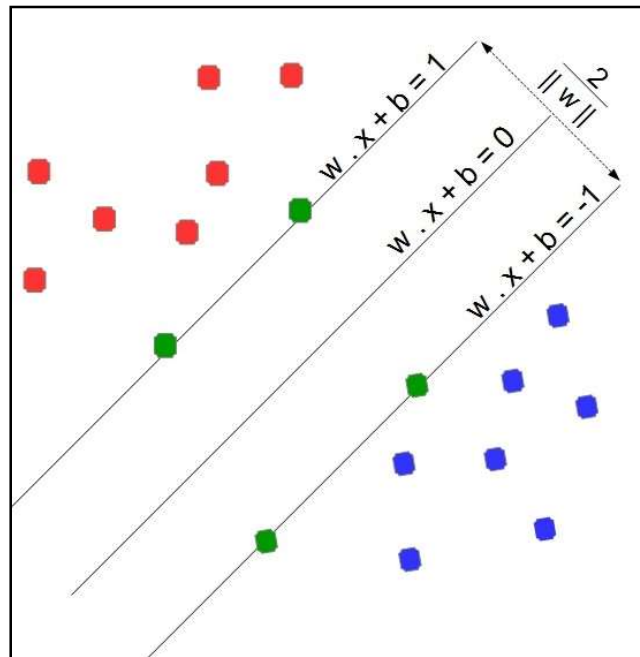
By using Bayes’ law, the probability of x belonging to category c_i can be found, which, in the context of this research is C is either spam or not spam by calculating the posterior probability of $P(C_i|X)$ by knowing $P(C_i)$, $P(X)$ and $P(X|C)$, where $P(C_i)$ is the prior probability of C_i and $P(X)$ is the probability of finding the feature’s value X , while $P(x|C_i)$ is the likelihood of an existing certain feature X in category C_i . Calculating the

likelihood depends on the probability distribution of X . X as a vector of features could contain variant types such as words, numbers or categorical variables. For this, different models may be used to represent X , for example, a multinomial model for discrete variables and a Gaussian one for continuous variables. The prior probability $P(C_i)$ is calculated by dividing the number of documents D (i.e. tweet, email, web page) that is classified as C_i by the total number of training data.

The assumption that features are independent of each other is not always valid; however, studies [71] generally report that the NB classifier works well with dependent and independent feature sets.

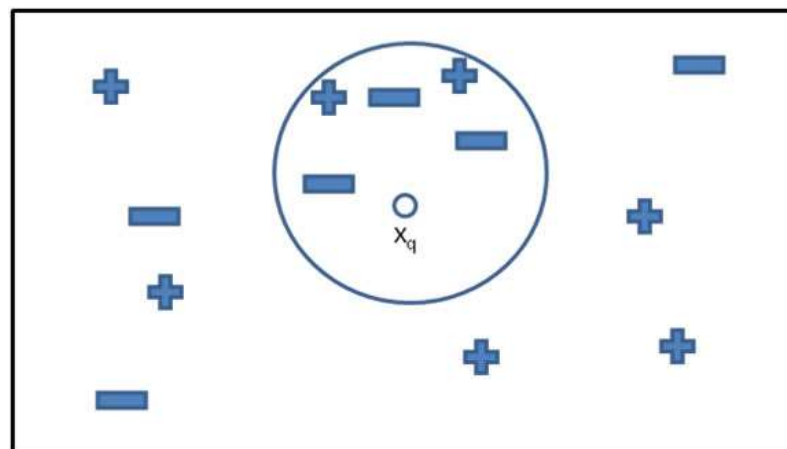
An SVM is a supervised learning algorithm that has been used by researchers on many classification problems. An SVM maps the training dataset vectors in higher dimensional space and then tries to find an optimal separating line (hyperplane) by dividing the vectors into classes. The optimal hyperplane is the hyperplane with the maximum margin; this is done by finding the maximum of $(\frac{2}{||w||})$ that separates the training data classes. Support vectors are the point in the training data that are needed for determining the maximum margin.

Figure 2.6 outlines SVM determining support vectors, which are the green points. These are needed for finding the maximum margin. As data may not be linearly separable, SVM performs a so-called kernel trick. If SVM cannot find a hyperplane to linearly separate a dataset in R^2 , it can transform the same data to a higher dimensional space where it can find a separable hyperplane.



**Figure 2.6 SVM trained with samples from two classes
green point are the support vectors that used to create the hyperplane to
separate the two classes.**

K-NN is a simple learning algorithm. It is a supervised learning method used in classification problems. K-NN maps the training input feature vectors $X = \{x_1, x_2, \dots, x_n\}$ in n -dimensional space, then classifies new data based on the majority class for the k neighbours. k refers to the number of training samples closest to the point of entry.



**Figure 2.7 K-NN example with $k=5$
As the circle represents unknown class data need to be predicted.**

Figure 2.7 shows an example where $k=5$ K-NN, which means that the K-NN finds five neighbours for the new point. There are multiple metrics for measuring the distance between points $X_1 = \{x_1, x_2, \dots, x_n\}$ and $X_2 = \{x_1, x_2, \dots, x_n\}$. Choosing the appropriate one is dependent on the type of data. There are multiple metrics to measure distance, such as Euclidean, which is the most commonly used; or the Hamming which is used in the case of categorical data. For example, the distance between X_1 and X_2 can be obtained by the equations in the Table 2.1 below:

Table 2.1 Examples of distance measurement metrics

<i>Continuous variables</i>		<i>Categorical variable</i>
<i>Euclidean</i>	<i>Manhattan</i>	<i>Hamming</i>
$\sqrt{\sum_{i=1}^n (x_{1i} - x_{2i})^2}$	$\sum_{i=1}^n (x_{1i} - x_{2i})$	$\sum_{i=1}^n x_{1i} - x_{2i} $ $x_1 = x_2 \Rightarrow 0$ $x_1 \neq x_2 \Rightarrow 1$

Determining the appropriate k parameter plays a key role in optimising the accuracy of the classifier. Choosing the inappropriate k value could reduce the accuracy of the classifier through data noise or merging group boundaries. Therefore, heuristic techniques are required to obtain an adequate k value. The simplicity of K-NN makes it one of the most common algorithms used in the spam classification problem.

The DT is a classification method that creates a tree structure called a decision tree, appendix C show example of a decision tree model. It breaks down training datasets into smaller groups to produce similarly labelled subsets and determines the most distinctive attributes that can enable the separation of the data into discrete

subsets. DT algorithms commonly rely on entropy and information gain to choose the highest impact to construct the decision tree [72]. Generally, entropy and information gain measure the homogeneity of a subset for each of the candidate's splitting features.

Building a tree requires identifying several key parameters [73], such as the number of features to use, the tree depth [74], and the minimum leaf size. The number of features is given by the dimensionality of the data that is used. The maximum tree depth is the maximum number of consecutive binary decisions that a decision tree is allowed to have. The minimum leaf size is the minimum number of data items that are expected to belong to the data subset associated with any leaf node of the decision tree. If the further splitting of the subset associated with a leaf node results in a leaf node that would have fewer data items associated with it than the minimum leaf size, the splitting does not take place and the node stays as a leaf node.

Each decision tree represents a tree of binary decisions that split subsets of the data into two. At each decision step, the most informative data feature is chosen (according to an informative or importance metric, e.g. information gain) for the subset of the data associated with the corresponding node of the decision tree. The dataset is split into two disjoint subsets according to this feature and two nodes are added to the decision tree, each having one of the two resulting subsets of the data associated with it. When a node is added to the tree, the node is initially a leaf node. If the data subset associated with the node is split further into two subsets, then the node becomes an internal node of the tree. The DT continues to split the training sample until it gets the same labelled dataset or there are no more features left for splitting. The fewer decisions that the model makes the better, as a complex tree could cause overfitting [75].

2.4.4 Ensemble learning algorithms

According to the literature (see Table 5.1), the ensemble learning method is one of the common algorithms and has the best performance. Ensemble learning aims to build machine learning models with better performance by combining several models. In general, researchers [76] have shown that combining several models is more likely to get better prediction than a single model. Many of the recent data science competitions have been won using ensemble method algorithms such as random forest (RF) and XGBoost¹¹ [77], [78].

Bootstrap aggregating (bagging) is an ensemble method that uses a collection of bootstrap data samples to fit multiple models usually from the same algorithm family, such as decision trees [79]. Fitting several models based on different views of the main dataset and then averaging their predictions helps to reduce the instability of the predictions. The bootstrapping method of sampling work involves getting random samples from the original dataset with replacement. Training on different bootstrap samples results in a different learning hypothesis on a certain instance, and by averaging those predictions or opinions, better overall performance can be achieved. This method demonstrates increased effectiveness on noisy data compared to using a singular model because of the random sampling of data [2].

The RF is an ensemble-based classifier, which means that it consists of a collection of sub-models that are used to make a joint decision. RF has several decision tree classifiers. These trees are built using a bootstrapping samples of the full training dataset, which results in potential differences between the trees, as the importance

¹¹ <https://github.com/dmlc/xgboost>

ranking of features may differ for different trees. The reliance on multiple decision trees to come up with a judgement makes RF classifiers more robust and less prone to overfitting than single decision trees and other non-ensemble methods [80].

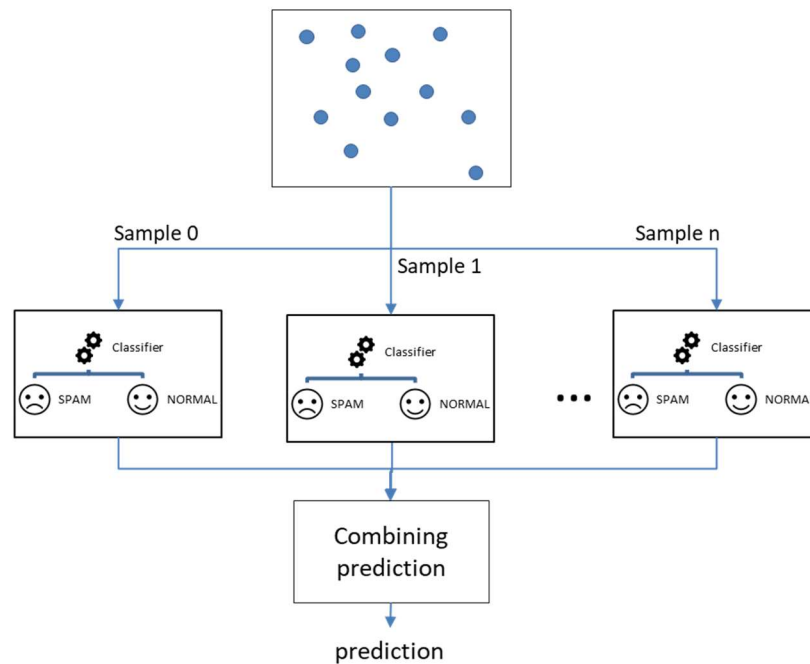


Figure 2.8 Random Forest generic mechanism.

The whole dataset is divided into n samples and each sample is used for building a singular DT. Then in the final stage, each model prediction is combined for the final prediction

RF is one of the well-known examples of a bagging method [79]. Bagging learning methods generally work by having multiple equally weighted base learners, and each learner is trained on a subset of the whole dataset. RF has an additional step to the traditional concept of bagging methods, which is selecting a subset of features instead of using the whole feature list. Predicting new unseen data in the case of RF is conducted by submitting the feature list of the unseen example to all m trees in the forest, getting the prediction results, and then creating a final prediction based on the average of all the trees' predictions. Compared to a singular decision tree, RF is better at handling noisy data and less prone to overfitting.

Extremely randomised trees [81] is another example of a bagging method. This algorithm has a similar methodology to RF but with some differences in feature set selection and finding the optimal cut-off point. This algorithm differentiates itself from RF as it does not calculate the best feature to be the split node or the split value of the selected features. Therefore, the term ‘extremely random’ refers to the selection of features and the cut-off point.

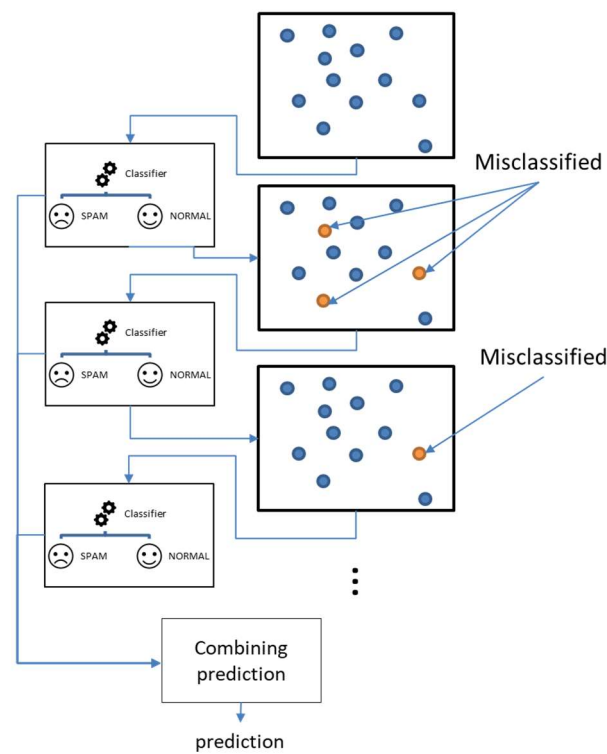


Figure 2.9 Boosting general mechanism
the whole dataset used in all n iterations of the training of boosting models

Boosting is an ensemble learning method that aims to improve the performance of a weak model by giving weight to the misclassified instances [82]. A weak model in the context of boosting learning means a model that has no previous guides about the data, so its performance more likely to be low. Boosting works by repeatedly training this weak model on the same training dataset, but in every iteration, the algorithm adds

extra weight to the examples that the model could not classify correctly. The final classifier is produced by a series of enhancements and adjustments on the first weak learner to make an ensemble model that is likely to give higher performance [83].

Unlike bagging, boosting algorithms do not bootstrap samples from the dataset or any kind of sampling the dataset. Boosting algorithms use the whole dataset for training but training examples are adjusted in every iteration. Every time the model is trained on the dataset, it evaluates itself and increases the weight of the misclassified examples, then passes these back into the model; the number of iterations being specified by the user is based on acceptable performance levels. This focus on misclassified examples makes boosting one of the best ensemble learning methods. Although this technique could show robustness to classify difficult examples, the performance will decrease dramatically when noisy or misclassified examples exist in the dataset, as boosting algorithms will try to weight highly noisy examples to try to fit the model.

The Adaptive boosting algorithm (AdaBoost) is another boosting-based concept. This is an algorithm that starts with a weak learner trained on a dataset and then the output is evaluated, giving more weight to the misclassified examples. The number of iterations of training and weighting misclassified examples is specified by the user. Finding the right values for these hyper-parameters is done by using tuning methods such as MDA and grid search. Compared to RF, this method can show better results depending on the dataset used. Small and noisy datasets are better fitted by RF than AdaBoost, as it is prone to give too much weight to the noisy examples [79].

eXtreme gradient boosting, known as XGBoost [78], is a new implementation of the gradient boosting trees algorithm. Enhancements include producing improved

learning or tree learning algorithms and making it faster and better in terms of scalability. Due to its high performance and ability to work in a distributed environment, it has become popular in many data science competitions.¹² The library is open source and well documented,¹³ and have supported and implemented for multiple programming languages (i.e. R language and Python).

Generally, the main difference between boosting and bagging is that bagging uses different bootstrapped samples to train several models and then applies equally weighted voting. Boosting works by training one model several times on all datasets but adjusts misclassified samples every time. Voting in boosting is weighted based on the performance of the model at every learning iteration. These differences make boosting and bagging quite different in their learning hypothesis and predictions.

2.5 Features used to build spam classifiers

This section provides a review of related studies on building machine learning-based spam detection models. The studies are classified by the type/source of features used in the machine learning method.

Features intended to appear in the context of this research are all traits (i.e. a URL points to a blacklisted domain) that are found in a tweet and attachments that increase the probability of correct classification as spam or non-spam of the tweet. Spammers try to keep their presence in social networks free from any suspicious behaviour as far as possible to avoid being detected [84][85]. So, researchers are keen to use features that cannot easily be disguised by spammers. Spammers use deceptive methods to

¹² <https://www.kaggle.com/competitions>

¹³ <https://xgboost.readthedocs.io/en/latest/>

normalise their behaviour in OSNs. The role of the researcher is to discover traits that reveal the spam content or at least restrict the activity of the spammers.

The aim of the security investigation is to find the best group of features that make sense in the security context and have strong distinguishing power when building the machine learning model. More time was invested to validate a group of features that were used in previous studies [86] [45] [87], such as Twitter metadata and web content. However, in the security context, features could lose their power due to the change in techniques that spammers used to deploy their spamming campaigns. This problem is called feature drift or fabrication, where spammers make content more benign by tweaking content spam features [88], [89]. Consequently, features adopted from previous studies need to be validated and features based on this investigation, such as page response to click actions or shortest domain age in redirection chain, need to be tested.

The efforts of researchers in the field of detecting spam content in OSNs have not been oriented towards finding novel classification methods as the majority of the previous studies aimed to find a novel feature set. Moreover, because there is no benchmark dataset and even no standard of labelling, researcher contributions varied in each step, starting by collecting and labelling dataset, features extraction/selection and method used to build models. In this section, related work considering the features used to build the detection models will be reviewed.

2.5.1 Twitter accounts and content features

Account-based heuristics use features of user behaviour to derive decisions about tweets. The method facilitates the process of observing suspicious behaviour seen in fake or infected accounts in OSNs [56]. Researchers have to observe real spam

content to know what features and characteristics to depend on in order to differentiate suspicious accounts from normal ones [90]. Researchers assess observations to come up with the most relevant set of salient features. Fake accounts' functions differ in OSNs, and these differences are reflected in their posting pattern and/or profile properties. Researchers have claimed to identify automated and suspicious behaviour through certain features [9] that are gathered from users' profiles and/or their published content.

Table 2.2 shows the features researchers [90]–[93] used in their machine learning methods. The first column shows features collected from Twitter user accounts and the second column presents a group of properties that could be found in the tweets' content.

Table 2.2 List of Twitter features adopted by previous studies [90]–[93]

<i>User features</i>	<i>Studies</i>	<i>Content features</i>	<i>Studies</i>
Length of profile name	[94]	Number of tweets posted per day	[9][95]
Length of profile description	[90] [95]	Number of tweets posted	[9]
Number of followings	[9] [95]	Retweet count	[92]
Number of followers	[9] [95]	Tweet content similarity	[45][96]
Account age (days)	[9] [64] [92]	Mentions	[97][96][64]
Ratio of number of followings and followers	[36] [9]	Tweet language	[95]
Real name	[98]	Number of hashtags	[9][64]
User verified	[99]	Number of URL links	[9][64]

All the characteristics mentioned in Table 2.2 can be collected using the Twitter stream API [100], which is provided by Twitter to allow researchers to access random portions of tweet feeds and profiles' metadata. Researchers use subsets of the

mentioned features to build classifiers based on the selection algorithm they have used. Several studies [9][36][96] have agreed that fake accounts tend to follow more accounts than they have followers for their own account. The analysis of this noticeable difference due to fraudulent accounts does not attract the attention of genuine users, and to attract them, they follow, mention, and favourite them. The high ratio of URLs in user content is also an indicator of spamming activities. One of the features that has been focused on by researchers is content/URL similarity. It is obvious that spam campaigns are run by a large number of accounts; based on this fact, many researchers have attempted to detect similar content in OSNs [101][102][36]. Stringhini [45] employed similarity in users' content and attached URLs to cluster users that might be deploying a spam campaign. Analysis was required to identify compromised accounts that are involved in this campaign. By building a behavioural profile of each suspected account, if the content is not consistent with the user's behaviour, then the user is identified as having a compromised account. Although this method could be useful to cluster tweet text content, in terms of URLs, spammers could evade original URLs using multiple shortened URLs.

Cresci *et al.* [103] used 49 distinct features derived from Twitter-only lightweight features. Lightweight features are those that do not require complicated pre-processing operations or significant resources to be extracted, e.g. age of account and number of followers. Chen *et al.* [86] extracted 12 lightweight features and build six different classification algorithms – BayesNet, NB, DT (C4.5), k-NN, SVMs and RF – which are ordered here in terms of their F-measure scores. RF had the highest performance at 93.6 per cent in the F-measure using an evenly distributed dataset. In the case of highly imbalanced datasets, however, such as a 1:19 ratio for spam/non-

spam, the performance dropped to 56.6 per cent. Classifiers that have been trained on imbalanced data are more likely to be subject to bias by majority class results.

McCord *et al.* [104] used lightweight features to classify user content into spam or normal. The features were extracted from Twitter account information (e.g. number of followers/following) and tweet content information, such as the number of mentions and hashtags. Similar to previous studies, the authors used several machine learning algorithms to compare their performance. The algorithms used were RF, SVM, NB, and k-NN. The authors found that the RF algorithm achieved the highest precision (95.7 per cent) and F-measure (0.957).

Although this is a passive detection method, researchers were able to detect fake/infected accounts or spam tweets that had not been discovered before, providing an advantage over the blacklists technique. The spam detection process requires it to be near to real time to stop spam from spreading on real-time content-sharing platforms such as Twitter and Facebook. Time is necessary for this technique to study the characteristics of an account or build a profile of behaviour.

There are some potential obstacles to the effective implementation of this method; spammers have ways to disguise the features that would raise suspicion of their fraudulent accounts. For example, examining the ‘following ratio’ feature, which has been used with previous notable studies [36] [9], becomes ineffective. Spammers start unfollowing those who do not follow them back; this is how they sustain the following rate as average [105]. Alternatively, spammers overcome the ratio of following to followers by making their ‘fake’ accounts connect with other fake accounts, increasing the follower and following numbers [106]. URL ratio is another indicator that has been evaded by some smart social bots. By monitoring real spam accounts, it has been found

that spamming accounts tweeting content with links pointing to malicious pages are also tweeting genuine tweets at the same average rate to tamp down suspicion aroused due to a high URL ratio.

The HSpam14 [107] dataset contains 14 million tweets that are labelled as spam and ham. The dataset labelling was conducted using several methods (heuristics, clustering and manual). These methods generated approximately 3.3 million spam and 10.7 ham examples. The main features that can be extracted from this dataset are Twitter-based features such as tweet content (text, hashtags, mentions and URLs). The study focused on spam injection on hashtags, as spammers seek to gain a bigger audience by attaching popular hashtags to their content. Studies [107], [108] that rely on only text only and twitter metadata and does extract features from attached URLs (domain WHOIS and webpage content) have the advantage of a larger dataset can create, but in the same time, they could miss essential spam traits in the attached URLs.

Compounding the difficulty of detecting anomalies in users' behaviour and characteristics are social bots, which are becoming smarter over time, adapting, and continuing to improve in simulating genuine users. Spambots have helped to overcome many effective features in a spam detection classifier. For example, by analysing text content that researchers use to cluster the tweets and try to find the spam campaign, spammers now use some machine learning algorithms to produce and paraphrase text to bypass string matching/clustering and blacklists [109].

2.5.2 URL, hosting and web page features

Researchers' work on most detection techniques is based upon discovering attributes or signs that help to detect malicious content in OSNs. Spammers try to avoid displaying any abnormal features so as not to be marked as suspicious. Consequently,

security researchers have expanded the domain of features to cover new aspects that have not been studied before. To overcome the previous techniques' weaknesses, researchers aim to make detection near to real time by finding traits that can be considered as conclusive evidence. Working with this technique does not require complicated historical behavioural study or detecting anomalies in users' behaviour; it involves dealing with the features that can be extracted from attached links [45]. The following table contains common attributes used by researchers [56][110] that serve to detect malicious URLs, and the table is divided into three types of features. The first relates to the URL's lexical features, the second involves the data that can be extracted from the website hosting server, and the last one refers to the web page content traits to which the URL refers.

Table 2.3 List of lexical, host-based and page-content features [56][110]

<i>Lexical</i>	<i>Host-based</i>	<i>Page-content</i>
Hostname	WHOIS Info	Popup messages
Primary domain	Server IP address	HTML Content
Path tokens	Geographic	JavaScript events
Sub-domains	IP hosting	Page screenshots

Although there is no feature that provides a 100 per cent accuracy rate for detecting malicious URLs, researchers studying these many features collected from various sources can give the overall probability and an indication of the existence of malicious content. Researchers [110][111] have found that fraudsters have ways to lure users to click on links that are designed to be similar to the websites they trust or use.

Table 2.4 Example of phishing URLs [112]

<i>Example of Phishing URLs</i>
http://login.paypal.com.nedgy.com
www.secure-paypal.com

Table 2.4 shows two examples of phishing URLs that many internet users thought were PayPal, which is an online electronic banking service. Blum *et al.* [111] worked on the lexical features of URLs by splitting them into three sections – protocol, domain and path – and try to detect the lexical features. The advantage of this detection method is the lightweight quality and speed of implementation without the need for the complexities of server information or page content. However, Thomas [15] expanded upon sources of features to include lexical hosting information and web page content, which has made it a comprehensive system. Feature extraction was done from multi-sources, such as web browsers, domain name system (DNS) resolvers, and IP analysis; most of the delay occurs when crawling URLs are using browsers as a service technique. The time required to analyse all the resources attached to a web page (i.e. JavaScript and CSS, images and web page screenshot) was an obstruction for researchers. For example, researchers have been trying [113][114] to detect phishing sites by finding visual similarities of web page screenshots of phishing and legitimate sites; however, the high computation resource and time required for this procedure was not commensurate with the nature of the big data that OSNs are dealing with. The primary restricting factor in this type of work is the high cost and time required, especially when it applies to real-time communication like that in platforms such as social networks.

Gupta *et al.* [87] proposed a mechanism to identify malicious URLs shortened by the bit.ly shortening service in particular. This shortening service is often used by spammers, who automatically generate malicious short URLs and spread them using their fake accounts on Twitter. Gupta *et al.* [79] built three models (NB, DT and RF) based on features of the landing page's domain information (WHOIS) and bit.ly features such as link creating hour and link clicks statistics information. The models were compared in terms of their classification performance. The authors reported that the RF classifiers showed the best performance for the considered data.

The deeper and specific features needed the greater number of resources to extract. A study has gone beyond monitoring web browsers by monitoring local system behaviours [115]. Burnap *et al.* used features of this type to detect malicious URLs. They deployed a high-interaction honeynet¹⁴ to collect system state changes, such as the sending/receiving packets and CPU usage. The training dataset contained 2,000 examples with a 1:1 ratio for spam/non-spam. Ten attributes were used to build a classifier that reflected system status changes after opening the tweet's URL. Burnap *et al.* [80] investigated the shortest time required to give a preliminary warning of the existence of malicious content in a specific URL. The best result was reported for a multilayer perceptron (MLP) using features acquired after 210 seconds (0.723 in the F-measure metric). The features used by Burnap *et al.* [80] require complex data analysis; however, they make it difficult for spammer sites to disguise their true nature.

The advantage of this focused group of features (system behaviour features) is its ability to detect zero-hour spam content/URLs. Studies that do not consider

¹⁴ <https://www.honeynet.org/>

analysing the historical profile of the accounts or not doing text similarity/clustering to detect spam campaigns are not waiting for enough information to make a decision. This technique could assess each entity individually regardless of other content that could be evaded and affect a classifier's decision. Furthermore, this technique can be used as a complementary technique for blacklist methods by updating its databases with the latest undiscovered spam.

In this thesis, the previous studies were investigated and clustered based on the group of features used for building their models. Each group of features has strengths and weaknesses in terms of its effectiveness or the resources required for extracting the features. It is also shown that features that are derived from sources such as domain, web page attachment and hosting cost spammers for disguising and renewing them when they are put on blacklists. Therefore, they are considered robust but cannot only be considered for discriminating spam and non-spam as spammers can renew hosting and domain names for each spam campaign. Therefore, the researchers needed to come up with group of features that can sense the behaviour of spamming accounts from social networks according to the way they lead users to the landing page. These combined features show different views of spamming activity – first the account spamming behaviour pattern, and then the pattern of content distributed.

2.6 Conclusion

There are several types of spam content; each has its indicators and there are some general traits that could group them. Moreover, the way of conducting spamming is varied among multiple techniques targeting several types of people in OSNs. Consequently, no one solution or detection method, algorithm or group of features can detect all the spam content in OSNs. Generally, previous studies were limited to the

spam type that was used and trained on, so there is a new trend to cover as many feature sets as possible to develop a comprehensive detection classifier. Furthermore, detection systems that were created using different methods need to be integrated in a way that they complement each other.

Although machine learning-based solutions are considered one of the methods with high potential to mitigate spam content in OSNs, most studies have built a model using a ground truth dataset, and eventually, the model became outdated and expired. One of the main important features that spam/suspicious content detection systems need to have is the ability to be adaptable to future methods and tricks used by spammers. Data sharing among research groups and security centres is needed so that these systems would be able to do retraining.

In the next chapter, the process of conducting the experiments in this thesis is outlined, and the experimental procedure, including the data collection and labelling and the process of evaluating and selecting the models, is discussed.

CHAPTER 3

Research Methodology

3.1 Background

To detect spam accounts in OSNs, along with spam content and activities, it is important to gain an understanding of the methods and techniques used to deploy spam. Second, a new countermeasure system to detect spam content in near to real time needs to be designed. To achieve the first goal, further investigation on current spam campaigns needs to be conducted and a computational workflow needs to be used to extract indicators and spatiotemporal features from real-time streaming tweets. To do this, data collection is the first phase to extract new, reliable spam-indicative features from the collected data. In this thesis, several investigations are conducted to assess features that are derived from various sources of data (i.e. web page content and redirection behaviour) to come up with a novel set of features. Unlike most of the previous studies, the scope of this research is not limited to one feature source. One of the contributions of the thesis is the data pre-processing experiments conducted, such as the methods used in feature collection and extraction, as extracting from multi-source data sources and types required several techniques such as web crawling, text processing and parsing, entity extraction, and dealing with unstructured and multi-languages data that was derived from WHOIS records of URLs' domain names. To evaluate how useful features are for discriminating, information gain and mean decrease accuracy (MDA) will be used as feature selection methods. Based on the selected features, a classifier needs to be built which can classify collected tweets as either spam or non-spam. In this research, the author plan to employ three popular

supervised classification algorithms and implement them using Scikit-learn¹⁵ (Python machine learning library). Furthermore, the author investigated what type of algorithm types that have been investigated by previous researchers [75], [86], [87] and suit the research problem and dataset in this thesis. The chosen algorithms have completely different ways of building models. Therefore, several different algorithms were chosen, RF, LR, NB and KNN, to determine which can give high performance and does not require high tuning effort. As the researchers' main aim is to build a dynamic detection system that performs the retraining process daily with new data, the less time needed for tuning and building the model the better. Later, the performance of each algorithm will be evaluated and comparisons will be made using the same ground truth dataset and selected set of features. To maintain efficacy, the classifier needs to be able to allow for periodic updates, extract new features and renew the training dataset so that it is adapted to new spamming strategies. Figure 3.1 illustrates the main phases of the workflow of the general experiments.

The proposed computational workflow can be divided into four consecutive phases. The first phase was data collection; the primary research data is from Twitter. Many researchers use the Twitter platform to collect data as most of Twitter's data is public. Moreover, Twitter provides APIs to facilitate the collection process. The aim of the second phase is to extract features from the collected data. Multiple feature sources will be considered in the extraction process, such as account, URLs attachments, and tweet content. Features will then be evaluated and ranked based on their discrimination power, with features that do not

¹⁵ scikit-learn: machine learning in Python, <http://scikit-learn.org/> [accessed November 2014]

play a role in the discrimination process being discarded. Furthermore, features will be verified regarding whether they were chosen by the authors' analysis or adapted from previous studies. As features lose their discriminating power for several reasons, analysis and validation are essential to check their effectiveness so that they can have a positive impact on building the model's accuracy rate.

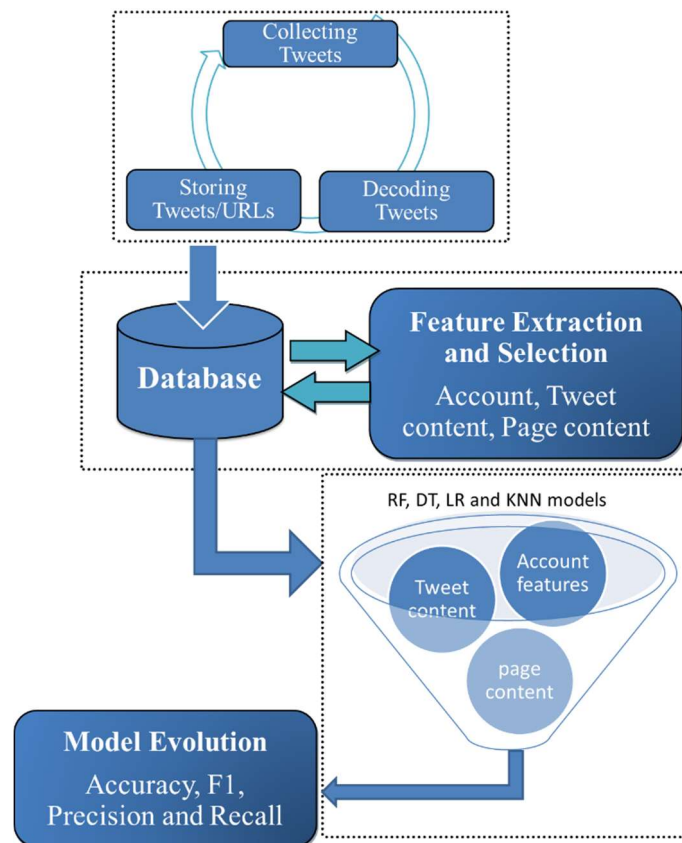


Figure 3.1 Key phases of the research methodology.

The collection phase is where tweets are imported into the database. This is followed by feature extraction and selection, and then building models and evaluating them.

In the third phase, a classifier was built based on the features that were selected in phase two. In this phase, features were subject to further analysis to determine which combination of features increased the classifier performance. Further investigation was performed to come up with high performance models using adequate settings of hyper-

parameters. Each set of parameter values or modifications applied to algorithms was analysed, and high-precision manually labelled ground truth data was used as a benchmark for the evaluation. Then, a system was developed that involved the selected features and the combination of methods and integrated into a usable interface. The final phase was where the overall system evaluations were applied. Two main criteria applied for the evaluation: system detection and features extraction speed. To measure the classification efficiency of the model, evaluation criteria adopted by previous relevant studies [116][117][118][9], such as recall, precision, accuracy, and F-measure, were used.

3.2 Data sources and labelling methods

This section explains how the data collection process was conducted and how the ground truth dataset of malicious and benign tweets was built. The data collection involved three steps: (i) collecting tweets that have URLs; (ii) crawling each URL using native web browsers controlled by Selenium WebDriver¹⁶ API; and (iii) labelling the URLs as malicious or benign.

Figure 3.1 shows that collecting data is the first thing to do to build the machine learning classifier. The main aim of this research is to build a classifier that can detect suspicious URLs spread over user-generated content websites. Several social network platforms suffer from a high percentage of spam content in their network. Facebook, Twitter, Pinterest and YouTube could be used as sources from which to collect data; however, their regulations and policies vary. In this thesis, the source used to collect

¹⁶ <https://www.seleniumhq.org/projects/webdriver/>

data is Twitter, although the study can be applied on any other social network that accepts users' text data with attached URLs.

3.2.1 Data collection process

Twitter has several connection channel APIs to obtain data from, such as real-time tweets, Ads API, Search API, Direct Message API, and filter real-time tweets (public stream API). Since the focus was on collecting new real-time data, public stream API (standard) and Twitter's public streaming API were chosen for tweet collection that give access to 1 per cent of the total stream [115]. A Python script was written to connect to the Twitter stream API and retrieve tweets that contained at least one URL. This returned a random fraction sample of new tweets to the developer in JavaScript Object Notation (JSON) format (see Appendix A). The only filtering rule used are that the tweet language should be English and there must be at least one URL attached to the tweet. After the collection software retrieved tweets and the above two conditions were applied to the stream tweets, the tweets were stored in a MongoDB¹⁷ database. MongoDB is a NoSQL database in which each row is a key and a document. The key is the unique ID that usually is auto-generated by MongoDB and each document is not required to follow a unified schema. NoSQL is perfect for data that is non-structured [119], which means that each document could have different fields and lengths, for example. This is different from a relational database when data should be validated and checked to match a certain schema to be inserted into the database. The dataset (tweets and URLs) in this research is unstructured, for instance, a tweet could have

¹⁷ <https://www.mongodb.com/>

information fields for attached images (media) or hashtags, so the document will have more fields than another tweet that does not contain any media attachments.



Figure 3.2 General techniques used and their flow
the flow represents the sequence of techniques used in processing incoming data till storing it in the database

Figure 3.3 shows the stage of collecting tweets, starting by connecting to the Twitter stream API, then applying the two filter rules specified, which are English tweets and tweets should have at least one URL attached. Around two million tweets were collected in the study from mid-2015 to mid-2018; however, scraping and extracting features from URLs attached to tweets required more time than obtaining data from tweets and storing it in the database. Therefore, tweets were stored in MongoDB as the first step, then another developed software solution was used for analysing and extracting URLs attached to tweets.

3.2.2 Data labelling process

When building a supervised machine learning model, a labelled dataset is needed. In this study, this meant labelling each tweet in the dataset as either ‘spam’ or as a ‘normal’ tweet. A ground truth dataset should be highly accurate and reliable enough to build supervised machine learning classifiers.

The lack of standard benchmark datasets in certain domains often forces researchers to build their own training datasets. Labelling a dataset is a challenging and time-

consuming, since there are no standard methods to follow. Researchers in this domain have used several ways to build the ground truth dataset for training their models. For example [87], [120] used third-party blacklist services such as VirusTotal and Google Safe Browsing to label tweets that contained blacklisted URLs as spam/malicious tweet. VirusTotal provides an API for retrieving information about URLs using up to 50 reputable online blacklists, such as Google Safe Browsing (Google), Bitdefender, Dr.Web Link Scanner, Kaspersky URL Advisor (Kaspersky), PhishTank (OpenDNS), Spam404, and Trend Micro Site Safety Centre (Trend Micro¹⁸).

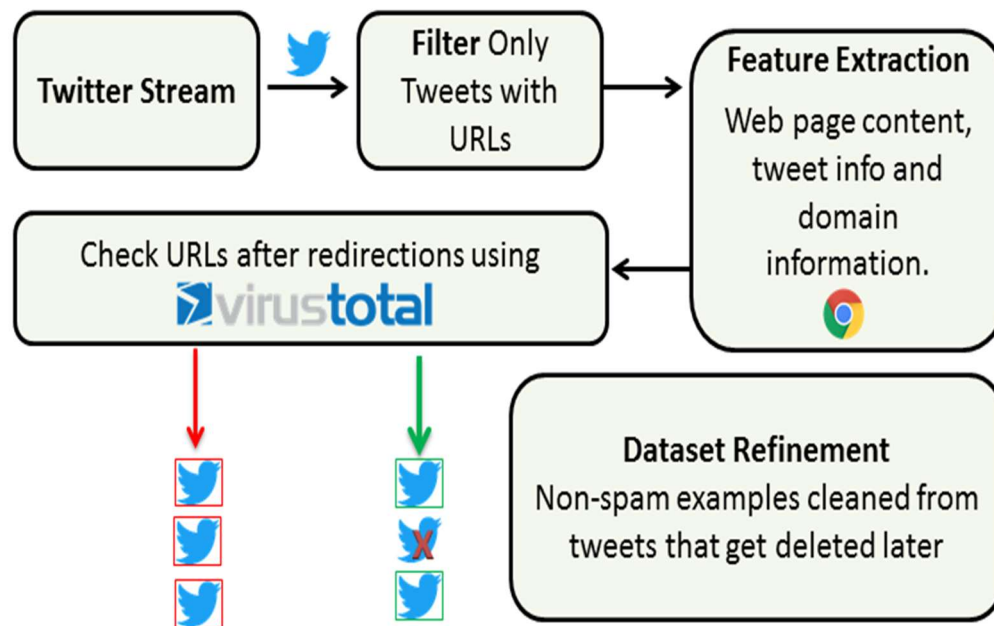


Figure 3.3 Data collection and features extraction workflow
crossed tweets represents tweet that got deleted by twitter

¹⁸ <https://global.sitesafety.trendmicro.com/>

Other researchers have used Twitter's own spam detection system for labelling [93], [121]. However, this is often delayed, as when accounts are deleted or suspended (for spam), the tweets that were posted from them can be labelled as spam. According to Twitter¹⁹, it cannot always be guaranteed that the reason for the suspension is related to spam, as there are several abusive activities that it acts upon. However, many studies indicate that the suspension is more often due to spamming activities [122]–[124]. Using blacklists and twitter suspension for labelling could help save time and effort for researchers compared to manual labelling. By reading the content of each tweet and browsing any URLs included, manual labelling produces more accurate datasets.

In this study, two datasets were collected; the first dataset (DS1) was used for the preliminary experiments and the second one was collected with a different labelling mechanism and features. The DS1 labelling mechanism relied on Twitter suspension and VirusTotal, so any tweet was considered that was deleted because of account suspension or had any attached URL that existed in the VirusTotal database as a spam tweet. This method saved time and effort, so it was possible to create a 15k ground truth dataset divided into 12k non-spam examples and 3k suspicious examples, which ranged from malware, phishing, scam pages, and overloaded ads to low-quality web pages. The dataset was also validated periodically using the two methods mentioned above, as some spam URLs required longer to be blacklisted or deleted by Twitter.

¹⁹ <https://help.twitter.com/en/managing-your-account/suspended-twitter-accounts>

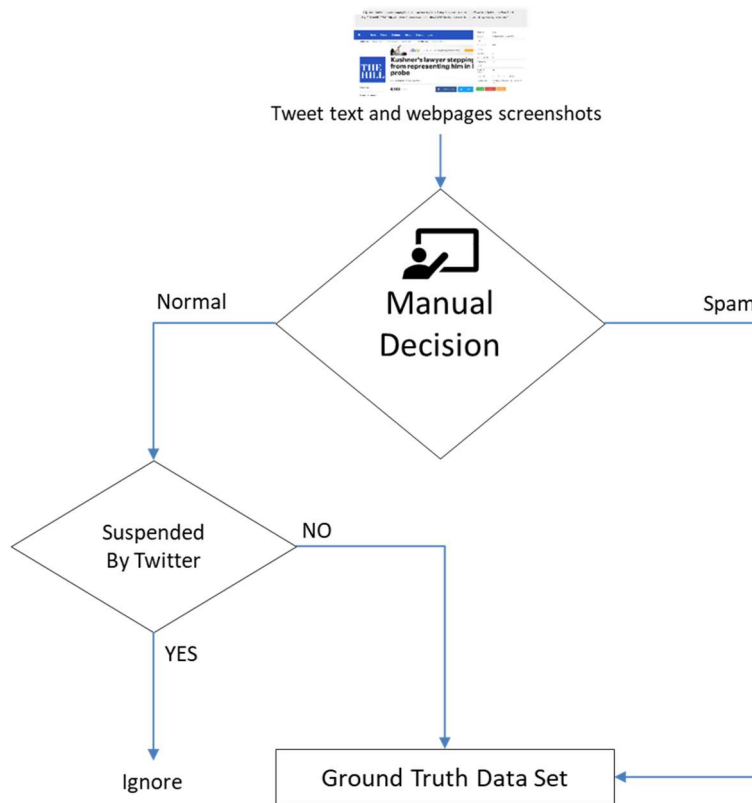


Figure 3.4 Labelling method used to build DS2 ground truth dataset, that involved extra manual validation

The second dataset (DS2) was labelled using a different mechanism. It was manually labelled by the researcher and refined using Twitter’s account suspension and VirusTotal to add a further refinement process to the dataset. The DS2 labelling process is illustrated in Figure 3.4. The 6k tweets that were manually labelled were reduced to 4k during the process due to manually removing spam labelled tweets which were deleted by Twitter or removing duplicate examples. The dataset was then checked using the Twitter suspension information, so every positive labelled tweet (spam tweet) was validated. It is assumed that for negative examples (non-spam), their source must stay active and online. Although the additional validation stage limited the size of the training dataset, it was considered important for the quality of the data, which is

essential for building valid machine learning models. Consequently, a ground truth dataset of 4112 examples was created, with nearly balanced classes as there are 2195 normal tweets and 1917 spam. Class imbalance in this context of study is common [125] due to the difficulties in detecting or manually labelling suspicious URLs.

DS2 tweets that contained URLs were labelled using a labelling tool developed specifically for this purpose to help researchers examine each screenshot and meta information to make a decision. Figure 3.5 shows the process of reviewing what was done manually by the author, who browsed the tweets using a bespoke program written in Python and Flask (web framework for Python) which showed the tweet content, tweet metadata (i.e. followers and account age) and domain/page features (i.e. number of windows opened and domain age). It also showed viewing pages' screenshots on the labelling tool page so that the features and screenshots could help the researcher to make the decision to label a tweet as spam or non-spam.

T @thehill: Kushner's lawyer stepping back from representing him in Trump-Russia probe; report <https://t.co/DapOKjndvV> <https://t.co/Grl3nl...>
 ['http://hill.cm/NFgGBJ5', 'http://thehill.com/homenews/administration/342100-kushners-lawyer-dropping-out-of-representing-him-report']

Features	Value
Record Id	5888976586238c1eb41b3273
Hots	1
DomainAge	8082
windows	1
alertsmsg	0
spammy words	4
NoofBILinks	13
userid	11111111111111111111
number of followers	69
created at	Sat Oct 22 03:48:00 +0000 2016
MLsuspicious	normal0.9754056488341183 XGB results [0.]

Normal Suspicious Not Sure

Figure 3.5 Labelling tool developed and used in labelling the dataset
In the bottom right corner of the screen shot image there are three buttons:
green (normal), red (spam) and yellow (unsure)

Table 3.1 Comparison between DS1 and DS2

	<i>Dataset 1</i>	<i>Dataset 2</i>
<i>Abbreviation</i>	DS1	DS2
<i>Number of Features</i>	34	45
<i>Features Categories</i>	<ul style="list-style-type: none"> • Twitter information • WHOIS (Domain Age only) • No text features • Focused only on landing page 	<ul style="list-style-type: none"> • WHOIS (3 Features) • Text analysis • Get features from all pages opened while/during reaching landing page
<i>Noise/Duplicates</i>	Unique tweets but there is a percentage of duplicate web page	No duplicates
<i>Labelling process</i>	Twitter suspension	Manual + Twitter suspension + VT
<i>Data time range</i>	Mid 2015 – end of 2016	End of 2016 – May 2018
<i>Pros</i>	<ul style="list-style-type: none"> • Easy to label • Larger dataset • Features low cost to extract 	<ul style="list-style-type: none"> • More accurate and less noisy • Features are more reliable, based on features ranking method.
<i>Cons</i>	<ul style="list-style-type: none"> • Less Accurate due to the labelling process • Simple features (easier to fabricate) 	<ul style="list-style-type: none"> • Smaller size, due to the manual labelling process for validation • Required more computing resource for features extraction
<i>Size</i>	55739	4112
<i>Classes size</i>	<ul style="list-style-type: none"> • Normal: 29956 • Spam: 25783 	<ul style="list-style-type: none"> • Normal: 2195 • Spam: 1917

In the additional refinement stage, each benign example in the dataset, i.e. a tweet containing a URL that was not blacklisted, was checked to determine whether it had been deleted by Twitter, as this may indicate whether that tweet contained malicious URLs that are not on a blacklist. According to Twitter’s deletion rules²⁰, there are three major reasons to delete a user’s²¹ tweet: breaking copyright, abusive tweeting activity, and that it is spam from Twitter’s perspective. To check whether a tweet had been deleted, the Twitter Streaming API was used to retrieve a specific tweet (using its ID). If nothing was retrieved via the API, then it was considered as deleted. This procedure

²⁰ <https://support.twitter.com/articles/18311>

²¹ There is also the chance that the user deletes the tweet.

was repeated several times during data collection, with the last check carried out in December 2016 for DS1 and May 2018 for DS2. The differences between the two datasets are summarized in Table 3.1.

3.2.3 Data preparation and unbalance issue

Preparing data for training, validation and evaluation is an essential process since the author can use the whole dataset for training. Furthermore, several issues need to be settled before using the dataset, for example, removing noise data such as redundant and missing values. For this, tools such as Pandas²² were used to check duplication and remove records that have full or almost full duplicate records.

For training and validation, the stratified cross-validation method was used where the dataset was split into k folds, with each fold having approximately the same target class percentage of the whole dataset. For example, if k assigns to 10, then training will be conducted ten times, each time trained on nine green split samples (as shown in Figure 3.6) and then tested on the tenth sample. The overall cross-validation evaluation was calculated by obtaining the average of all the k folds. The cross-validation method helps in mitigating falling into the overfitting problem by using the whole dataset for training and evaluating so that the model will be more generalisable. As the general aim of building a machine learning model is to help in predicting (classification or regression) new unseen data, methods such as cross-validation will help in achieving a

²² <https://pandas.pydata.org/>

more reliable evaluated performance, as in cross-validation all the data is used for training and testing.

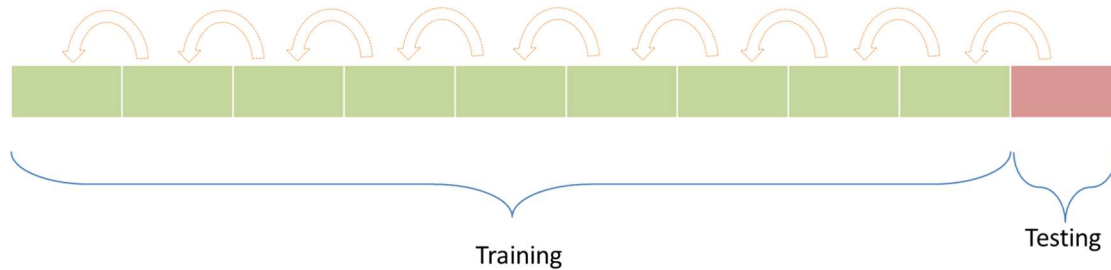


Figure 3.6 Cross-validation sampling method

On the other hand, in the context of spam detection in OSNs, many researchers have noted the problem of unbalanced datasets [126]. To solve this issue, in this study, the researchers performed undersampling of the negative examples (the dominant class) by removing the redundant and semi-redundant examples. Moreover, machine learning models that are better at handling class unbalance were chosen later, for example, ensemble methods that combine several classifiers demonstrating improved handling of unbalanced datasets [125]. Metrics that are also used for evaluation were chosen to support this issue. F-measure (F1) and area under the curve (AUC) are both more suited to handling class unbalance in their evaluation.

3.3 Feature extraction and engineering

Each row in the dataset contains several columns; each column is a feature. The better the features set describes the data points, the more distinguishable the data points will become, improving the classification performance in determining spam and non-spam tweets. Stored data was transferred into a features vector that a machine learning model can understand. A machine learning model use a vector of numbers, which each vector represents an example of input data point. Thus, a model uses these numbers to

make a classification decision. Features can be extracted from several data formats, such as text, HTML, images, and URL chains. The complexity of extracting features varies, with some requiring several processing stages to convert them into a format suitable for machine learning. For example, features contained in the Twitter metadata, such as number of followers, were easily accessed in the returned data from the Twitter Streaming API. However, to obtain the full domain WHOIS record, requests to the registry's WHOIS databases had to be made and then the response data had to be parsed to extract the relevant group of features needed. To extract features related to the web page content that was pointed to by tweets' URLs, it was necessary to use a headless browser, Selenium²³, to automate browsing behaviour and content download. As Selenium provides API-controlled native web browsers, the author opens all tweet attached URLs via a programming language (in this research, Python was used).

Table 3.2 Sources of features used in building machine learning models

<i>Source</i>	<i>Features</i>
Twitter network	Twitter profile and tweet features
Web browser	All web page source code
Web browser actions	JavaScript events occur while crawling website (popups windows, alerts, etc.)
URL behaviour	Number of redirections
Domain WHOIS information	Domain WHOIS record that contains information such as registrar and dates related to domain registration and expiration

Studies [127] and [37] have shown that using multiple sources of features could help to increase the classification power [58]. Therefore, in this study, features that were derived from several sources were explored, as shown in Table 3.2. The investigation

²³ <http://www.seleniumhq.org/>

was conducted by monitoring real spamming accounts on Twitter and studying their produced tweets. Moreover, browsing and analysing tweet-attached URLs was performed to gain an understanding of the tricks that spammers used. Features that were derived from previous studies' classifiers were also assessed.

3.3.1 Features used in building classifiers

In this section, a description of the features that were used is provided.

Twitter network:

Although recent studies have shown a decrease in the detection power of features that derived from Twitter, it was still possible to detect a percentage of attacks by using features of this type. The Twitter platform provided thousands of tweets in seconds of random real-time tweets; however, this rate could be lower depending on the filters used in the Twitter stream API. Due to the simplicity and high rate of tweets that Twitter offers to the public, many researcher have used Twitter as a primary data source for relevant studies [86][91][104]. Using their API, it was possible to retrieve records of information about users and tweets. Some Twitter metadata could be in numeric format, which is preferable for a machine learning model.

Table 3.3 Common features used in literature

<i>Feature</i>	<i>Features Description</i>	<i>Ref</i>
Account age	Number of days since account created	[86][91]
No. of followers	Number of accounts connected to this account	[86][91][104]
No. following	Number of accounts connected to this account	[86][91][104]
No. of user favourites	Number of favourite tweets	[86]
No. of tweets	Number of posted tweets	[86][91][2]

Table 3.3 shows examples of numerical Twitter features that have been commonly used in previous studies [86][91]. The advantage of using this type of lightweight features is that it does not require a high level of extraction and transformation.

User information:

Table 3.4 User information features

#	<i>Feature</i>
1	Ratio of age to number of tweets (Twitter)
2	User statuses count (Twitter)
3	User friends count (Twitter)
4	Account age (Twitter)
5	User followers count (Twitter)
6	User favorites count (Twitter)
7	User listed count (Twitter)
8	User name length (Twitter)
9	User name digits (Twitter)
10	User name signs (Twitter)
11	Default profile image (Twitter)
12	Is user account verified? (Twitter)
13	Is user account protected? (Twitter)

Features such as created date (account age in days), number of tweets that account generated since it was created, and number of followers and following accounts can represent the user who posted the tweet. Furthermore, account username and biography description text can be used as text features. New features can be created by combining features using arithmetical procedures such as feature no.1 in Table 3.4, which is the value of the division of number of tweets by account age. This ratio could indicate

account activity, as a spam account could have much higher activity compared to the average normal users. Feature engineering can be used to create another feature based on existing features, for example, [128] used the user reputation/fame feature, which is high when an account has higher followers count than following count.

Tweet metadata:

Table 3.5 Tweet features

#	<i>Feature</i>
1	Number of hashtags (Twitter)
2	Does tweet have media? (Twitter)
3	Number of mentions (Twitter)
4	Is tweet a reply tweet? (Twitter)
5	Number of URLs (Twitter)
6	Is user account geo-enabled? (Twitter)

Tweet text is not the only piece of data that can be obtained, as tweets can have many types of other metadata attached, such as hashtags, mentions, tweet type (reply, retweet and standard tweet), and, most important to this study, URLs. URLs are one of the features that have been examined in previous studies, such as [87][91], which investigated similarities in tweet text to find a spam campaign. This could be important for checking against a pre-trained text classifier that was trained on thousands of spam and genuine tweets.

Web page content and behaviour:

Going deeper to follow spammers to their end landing page where they want to lead targeted users is an essential process to understand what the content is that they want to spam about. To collect this content and extract relevant features, author automate the process of crawling all tweets' attached URLs and handle all further redirections that

happened. Interacting with landed web pages and resolving redirection could require building a smart crawler that is able to handle all these obstacles (i.e. HTTP and JavaScript redirection).

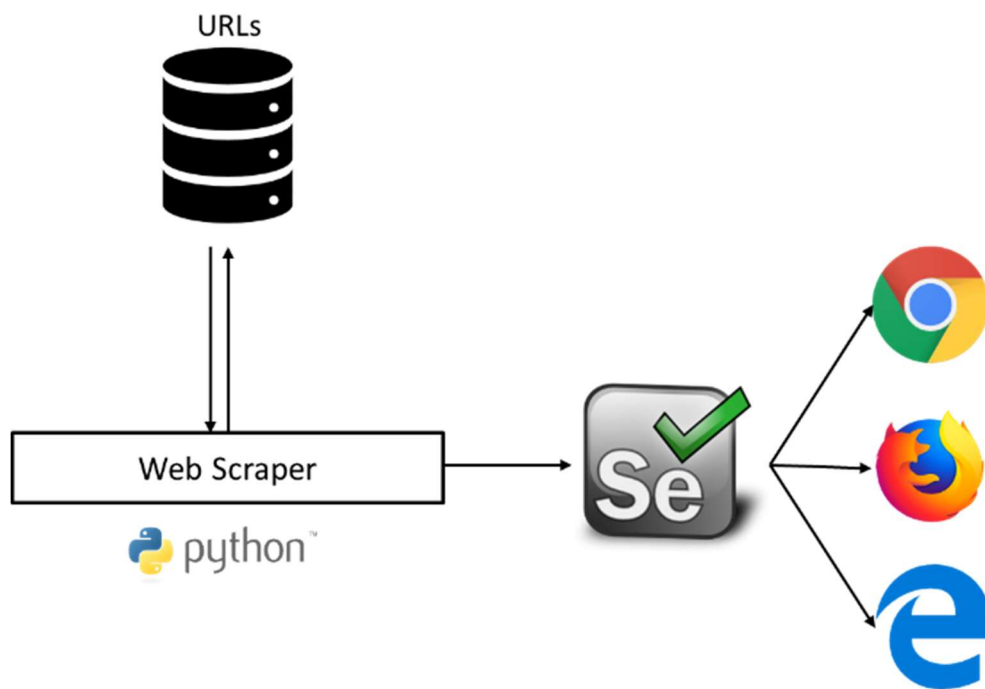


Figure 3.7 Selenium web drivers
web browsers logos represents the capability of selenium to use different software web drivers

For this purpose, the Selenium WebDriver API was used, which enabled researchers to have full monitoring of web browsers' actions and properties from loading to unloading the web page. The author assigned normal browsers (i.e. Firefox and Chrome) to open URLs and by using this library (Selenium), were able to gather page content and behaviour, such as the redirection hubs, and get the final landing web page. Figure 3.7 shows how the web scraper component exploits the Selenium driver to control web browsers and pass tweets' URLs to them. Finally, when browsers open URLs, all the web pages' data and behaviours are stored back to the database.

Table 3.6 Redirections' observation features and web page content

#	<i>Feature</i>
1	Number of external links (web page)
2	Number of links (web page)
3	Number of images (web page)
4	Number of dots in link
5	Ratio of words to external links (web page)
6	Number of input forms (web page)
7	Number of words (web page)
8	Link length (URL)
9	Number of link signs (web page)
10	Number of ad blocked links (web page)
11	Does link contain 'www' ²⁴ ?
12	Does web page have password input? (web page)
13	Link letters (URL)
14	Is https protocol used in URL? (URL)
15	Popup windows
16	Alert and dialog messages

A high-speed connected machine and high processing speed were used to retrieve all URLs in the dataset. A Python code connecting with the Selenium API was set up to control web browsers on a Core i7 32GB RAM machine to visit each tweet's URL to collect additional source data. Features were extracted from web pages that opened with and during opening of the original URL. Table 3.6 shows all features extracted from the process of crawling URLs. Features ranged from ready to use (i.e. number of links

²⁴ <http://www.yes-www.org/why-use-www/>

and number of images) to features that required further processing to be extracted, such as number of ad blocks.

Unlike previous studies, the features in this study were derived from pages' content and behaviour during page redirection until the redirection chain reaches the final landing page. As this study is based on real examples of spam content/URLs, some tricks used by spammers show spam content on the way to the landing page, then lead users to the legitimate web page. Using this trick helps them to overcome systems that rely on analysing the web page content of the landing page that URLs are pointing to. Therefore, this thesis presents the novel features that could help to detect such attacks and tricks used by spammers to bypass detection systems.

Redirection feature:

A user could open a link which could start with a benign and clean web page then suddenly be redirected to a harmful malicious web page. Redirection is one of the common tricks that spammers use to overcome detection systems that investigate the landed page's URL or content [93][87][129]. Moreover, requesting a page using a URL could be redirected once or more. Consequently, the researchers built a component that extracts the full redirection chain and stores all content and URLs that come through the redirection process. Redirection could be done through multiple techniques, such as HTTP header, JavaScript action or timers, so the author needed to use a smart tool that follows all redirections that happened. Features such as number of redirections could be a good indicator that something is not normal in the page. This motivated us to invest more in collecting data during page redirection, as some spammers inject malicious web pages in the middle of the redirection chain.

The software used for crawling and handling the redirection needs to be static for all URLs, since changing the network, device or location could change the redirection behaviour. For example, opening a URL from Iraq could lead to or get a different route to the landing page if open it from the UK. Therefore, changing the URL entry point in terms of the device or location could change the redirection route. Consequently, to have an unbiased training dataset, in this thesis the author used the same device and network for crawling all the dataset URLs.

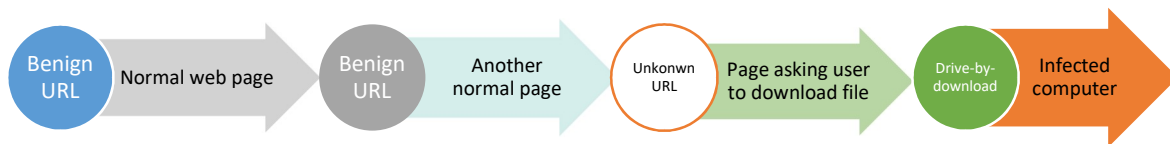


Figure 3.8 Malicious/spam content hidden content behind multiple redirected URLs
starting by un-blacklisted URL and ending with blacklisted one

Figure 3.8 shows an example of a drive-by download attack evaded in several redirection processes. As spammers can exploit public shortening services to conceal their suspicious URLs, if the user opens a URL that looks benign, then this URL is redirected to another server/domain and then to another page, and this page could contain scam content to try to convince the user to download software (i.e. Flash player, antivirus). This software could lead to the user's device being infected and controlled by an attacker/hacker.

Web page content features:

Once the developed extraction process reached the final landing web page, further analysis was conducted to try to extract features from its source code content. This

landing page should be the page that is intended to be reached by targeted OSN users. Therefore, the author focused on several features on this page, such as number of images, URLs, external URLs, and distinct domain name URLs. Some features, such as number of images, scripts and href tags, were extracted using text processing and HTML parsing functions. BeautifulSoup 4²⁵ (BS4) is a Python library used for parsing web page source code text data into HTML parsed code, so the author were able to do a search and pull data easily. All web pages' content was stored in the MongoDB database for later use in case it was necessary to extract further features from their source code.

Web page behaviour:

A web page comprises three main elements: HTML, Cascading Style Sheets (CSS), and JavaScript. HTML is a standard markup language used for creating the structure and content of web pages. CSS is used for describing the presentation of the structure and content of the page. JavaScript is responsible for the web page behaviour regarding events and interactions with users. Web page behavioural features are those which represent page events' load and unload page messages and popup windows. In the stage of collecting and extracting features, three events and actions were considered: page loading, mouse click on the HTML body tag, and page unloading. During these three actions, some web pages behaved differently, so the author wanted to determine how the majority of legitimate/normal and suspicious web pages behave in responding to these actions.

²⁵ <https://www.crummy.com/software/BeautifulSoup/>

Popup windows, iframes and alerts:

Web browsers have the ability to load another web page window/tab to users who clicked on the attached URL in a tweet. A web page using several techniques can show alerts, popup windows, and iframes. Loading another web page or open popup windows could be an unpleasant action, especially if there was no real need for it. Showing more content than the user is expecting from a web page is itself an annoying action [130]. Figure 3.9 shows an example of a popup window that contains a potential scam advertisement luring users to click.



Figure 3.9 Example of a popup window used for advertising

Some popup windows needed a trigger to be opened, for example, mouse movement, window loading, or closing or clicking on a certain part of the page or sometimes on any part of the page. The author is not suggesting that any web page that has popup windows is suspicious; however, this could be one of the features that could enhance the prediction performance of the classifier decision. As popup windows with annoying ads could be a sign of a low-quality web page, the author performed further investigation to study the existence of popup windows and suspicious web pages.

Again, the content of these popup windows' frames opened as a result of loading the original link and landing page that were stored in the database.

Excessive amounts of advertising:

As stated earlier, suspicious content in a page can be in many forms, one of which is pages that are only seeking to make money by having small, low-quality, clickbait content or even false news and the rest of the web page is annoying ad blocks. Some of the previous studies [64], [122] categorised web pages that expose users to excessive advertisement content that is either legal or illegal content as low-quality suspicious web pages. These ads could be injected directly into the main landed web page inside a div tag used specifically for advertisement content or the content could be loaded in an isolated iframe. Iframes are an HTML document loaded as popup content with an HTML document (parent document). Advertisers prefer using an iframe when delivering their advertised content to ensure that it appears in the same way as they designed it regardless of what the parent document design is [131].

Therefore, to obtain feature 10 from Table 3.6, which is number of ads blocked in a web page, the author needed to build a Python code using Adblock parser library that uses an updating list called Easylist, which includes a list of domains that are known for advertising or certain pattern ad codes can be found in the web page source code. By using Adblock and Easylist rules, the author was able to detect advertising blocks in the web page and the attached pages such as iframes and popup windows while loading/unloading the main web page.



Figure 3.10 Example of ads abusive web page
two column web page as both contains ad text and banners

Domain WHOIS information:

WHOIS is a protocol used for querying domain and internet protocol (IP) address registration information databases. Each domain has a record of information hosted in the DNS registrar server. This information is provided as plain text information presented under different syntax and headlines, and it could be in different languages, so parsing and extracting information is a challenge. The information could include registrar company and important dates such as when the domain was registered and updated and will expire. Domain owner name, registrar name and technical administrator contact address are usually stored in the domain WHOIS record. The WHOIS record is one of the sources of information to extract one of the top features, which is the creation date of the domain (domain age). Domain age is the number of days from when the domain was created to the day the tweet was created. The oldest

domain name was created on 15/03/1985, so it is 33 years old in 2018 (approximately 11,800 days), which belongs to symbolics.com²⁶. The lowest value for domain name could be one day, which means that a tweet with a URL attached has a domain name that was created on the day that the tweet was tweeted. Registrar name represent the name of the company that sells the domain to the registrant, who is the domain buyer.

Table 3.7 WHOIS information features

#	<i>Feature</i>
1	Domain age
2	Distinct domain name in redirection chain
3	Registrar name (text)
4	Number of valid fields in retrieved WHOIS data

As a part of the development of the system, a code was developed that parses domain WHOIS information and extracts the features described in above table.

The features mentioned in this chapter are essential in building a robust discriminative model. Features ranged from simple extracted features such as those derived from the Twitter stream API JSON data format to those that required pre-processing several times and handling all unexpected actions such as popup windows and alert messages in web page content features. In this thesis, the author used the term ‘lightweight features’ to represent features that did not require pre-processing work and the term ‘deep features’ for those that needed more effort and pre-processing. Based on investigations conducted on real spam examples, spammers can easily overcome

²⁶ <https://www.whois.com/whois/symbolics.com>

detection methods that rely on lightweight feature models by using shortened URLs and disguising text to make it more like a legitimate tweet. Overcoming this obstacle requires few resources and little cost; however, deep feature-based models cost spammers more to overcome as those models dig deeper into domain name and web page content or even analyse attachments. Buying thousands of fake accounts could cost around £10 and each account can reach thousands of OSNs users, but buying one domain name also costs around £10. Therefore, it costs spammers more to disguise features extracted from domain names and web pages. Consequently, choosing the right features could make a difference in the detection performance, so it requires an understanding of current spamming campaigns and attacks to understand how spam mechanisms and the industry work.

3.4 Model selection

In machine learning, there are three learning methods, supervised, semi-supervised and unsupervised, which researchers choose according to the nature of their research problem and dataset. Supervised learning is suitable in cases when there is prior knowledge about the labels and in cases when there is availability of a labelled dataset. Furthermore, the ratio of classes can have an impact on choosing the method. For example, cases of fraud transaction detection and network intrusion detection system data unbalance are considered as huge; therefore, the applied algorithm needs to be well analysed to ensure that the model is not only biased to the majority class. For instance, anomaly detection algorithms would be suitable for highly unbalanced datasets such as those that have very low positive examples. On the other hand, if there is no enough labelled dataset, semi-supervised or unsupervised learning algorithms are likely the optimal options.

In this thesis, the aim is to detect suspicious URLs that are spread over social networks (Twitter) which are labelled as either spam or not spam. Based on the nature of the problem and the availability of labelled spam/non-spam, the common learning method used in the literature for the spam detection problem is supervised machine learning algorithms [132], [133].

3.4.1 Model selection criteria

As shown earlier, supervised machine learning is what the author will use to build the spam detection classifier; however, there are several supervised machine learning algorithms. Model selection in the machine learning context is the process of choosing the best model among different models or even the same model using different hyper-parameters or feature sets. Model evaluation is not limited to classification performance only, as several characteristics can be used to assess models. These characteristics are outlined below.

Classification performance

Several metrics, such as accuracy, recall and precision, are used to assess classifier performance. Each evaluation metric has advantages based on the classification problem and dataset class ratio. In spam detection, the researchers need to build a classifier that has very high spam detection performance and very low false positives, since the effect of labelling a normal tweet as spam would annoy users more than seeing some spam content from time to time [49]. Therefore, precision needs to be a higher priority than recall. Moreover, class unbalance is a common issue, so using a metric such as accuracy can sometimes be misleading. For example, if there is a classifier that classifies every tweet as non-spam tweet and the dataset has a normal to

spam ratio of 10:1, in this case, the classifier accuracy will be 90 per cent. Consequently, evolution metric that suits the spam problem and dataset needed to be found. To evaluate the classifiers, the commonly used evaluation metrics of accuracy, precision, recall and F-score were used.

Table 3.8 Confusion matrix

<i>Dataset</i>		<i>Classifier Decision</i>	
		<i>Spam</i>	<i>Not Spam</i>
<i>True</i>	<i>Spam</i>	True Positive	False Negative
	<i>Not Spam</i>	False Positive	True Negative

- AUC represents the classifier's ability to detect classes. If the AUC is 1, this means that the classifier perfectly detects class labels, whereas 0.5 is equal to random selection. AUC performance metric has proved its ability in providing reliable performance even when dataset is imbalanced or cost sensitive problems dataset [134][135][136].
- Precision is the ratio of the true level of positive or negative detection of the classifier to overall test samples.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (2)$$

- The recall is the ratio of correct true positive classifier decisions to all the true positive examples in the test set.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \quad (3)$$

- F1 represents the previous metrics' precision and recall combined as follows:

$$\text{F1} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}) \quad (4)$$

Ease of understanding the model outputs

It can be difficult to understand how decisions in complex machine learning models are made. For example, deep learning models are considered to be one of the types of models that were proved to have high performance for many classification and regression problems. However, it is difficult to determine how a decision is made based on complex weight computing on several layers of neural networks. However, in a DT model, researchers can follow each case to see how the splitting until it reaches the final leaf, where the final probability is computed. Knowing which features and values impact the decision could a required feature in some machine learning-based systems (i.e. medical and military systems).

In the context of internet security, there is a trade-off problem between having transparent versus opaque models. Therefore, spammers are keen to extract the rules and internal decisions that happen in the model so that they can find ways to hidden features that would cause their content to be detected. Consequently, the model transparency can be a useful feature in less critical cases where no adversarial attacks are expected. However, this feature can be considered as a drawback or an advantage based on the task that the model is applied to.

Building and tuning

The increase in machine learning libraries and projects that are supported by highly regarded open source communities has enhanced the productivity of researchers.

Libraries such as Scikit-learn²⁷, TensorFlow²⁸ and XGBoost²⁹ have boosted the building of machine learning models and led researchers to focus on the area where those models are applied. However, researchers are required to have some knowledge of how the algorithms work and how to tune its hyper-parameters them.

Algorithms have parameters that the algorithm itself tunes during the training process, and there are hyper-parameters where the researcher needs to be involved and determine the hyper-parameter settings. Machine learning algorithms are varied in terms of the number of hyper-parameters and the complexity and effort required to find the best values and options. It generally requires time and resources to find the best configuration to build a model. Currently, there is a trend to make models that require less human involvement in tuning and selecting features, so the less human intervention required, the better.

Resources and costs

Choosing the best model also depends on the available resources and expertise. Therefore, when choosing an algorithm to build a spam detection classifier, it is necessary to study and understand the resources and expertise required to choose an adequate algorithm. For example, algorithms such as ensemble learning and deep learning algorithms could require high training computing resources, as ensemble learning could build thousands of learners during the training phase.

²⁷ <http://scikit-learn.org/>

²⁸ <https://www.tensorflow.org/>

²⁹ <https://github.com/dmlc/xgboost>

Consequently, the model selected in this study was chosen based on several preliminary experiments. The experiments show that the training time for the research dataset is acceptable for most supervised machine learning models due to the medium size of the data. The details and information relating to these experiments are presented in chapter 4.

3.5 Hyperparameter tuning

After choosing a classification algorithm to use to build the model, more time was invested in optimising the model's performance. Each model has internal parameters and external parameters, in other words, hyper-parameters. Internal parameters are computed internally during model training. For example, in the neural network model, during the training process the neural weight is assigned internally by the algorithm without human intervention. Alternatively, the researcher chooses the hyper-parameters for the model, for example, in the decision trees model the researcher assigns the maximum depth of the tree before the training process. So, hyper-parameters are not calculated through the model, but they are chosen by the modeller.

Finding the best combination of hyper-parameters can enhance and optimise classification performance. There are several methods to obtain the best values/options of parameters, such as mixing them up to come up with the best combination, which is called a grid search, ideal for a small data size. Random search, which basically does x random combinations of parameters then chooses the best performance, can be used when it is difficult to test all the parameter combinations due to either the huge number of options or the huge size of the dataset. Moreover, one of the methods that can be used in such cases is the Bayesian optimisation method, where another algorithm can be used for estimating and evaluating performance, and based on previous performance,

a better hyper-parameters setting can be set until a satisfactory state is reached. Tools such as MOE and Spearmint³⁰ can be useful in applying the Bayesian optimisation method on different models.

3.5.1 Feature selection methods

As mentioned in section 3.3, the higher distinguish power features are, the better the classification performance will be. The feature selection procedure was conducted in two stages. In the first stage, each feature in the original feature set was evaluated by applying several available feature selection metrics such as Chi square and information gain. In the second stage, the top k features were chosen which should have the highest discriminative power (i.e. domain age, account age, and number of redirects) [38]. Evaluating features, which is also known as the feature importance score, is an essential process to understand the dataset that used to build models. Moreover, it enables a researcher to distinguish between good features and irrelevant features. Eliminating redundant and noisy features could cause performance improvement [39]. There are three generic methods to for feature selection, such as the filter, embedded and wrapper method [40]. Several feature selection algorithms were applied to the dataset in this research. This thesis needed to validate the selected group of features developed or adopted from previous studies. All the experiments and results are presented in chapter four, section 4.4.

³⁰ <https://github.com/HIPS/Spearmint>

3.6 Discussions

This research aims to detect tweets that have implicit spam content in the tweet text or metadata and/or the content that tweets with attached URLs are pointing to. Since the research aims to mitigate the risks of opening suspicious URLs in OSNs, the researchers focused on tweets that contain attached URLs. As spammers could easily evade spam content behind very benign tweet text but the attached URL is pointing to a suspicious web page, in the analysis the author needed to go beyond analysing tweet text only.

Several steps are required to build a machine learning model to detect suspicious URLs spreading over social networks. The first step is having a high-quality dataset to distinguish features extracted. The researcher manually investigated hundreds of real spamming accounts to gain an understanding of the techniques and tricks used by spammers alongside methods used for evading spam web pages. Extracting features is one step in finding the best group of features that make sense in the security context and have strong distinguishing power when building the machine learning model. Based on the model, these features needed to be tested to determine whether they had a positive impact on the model's performance. This is due to the fact that in the security context, features could lose their power due to the change in techniques that spammers used to deploy their spamming campaigns.

Since machine learning models are very sensitive to the trained data, most of the researcher's effort goes into building a dataset and extracting and engineering features. Figure 3.11 shows the steps of the research methodology in building the spam classification model.

Starting by collecting dataset as the first step, then several method of feature extraction and crawling used on collected tweets and their attached URLs. Features extraction component can be grouped into twitter, reduction, web page content/behaviour and domain WHOIS record features.

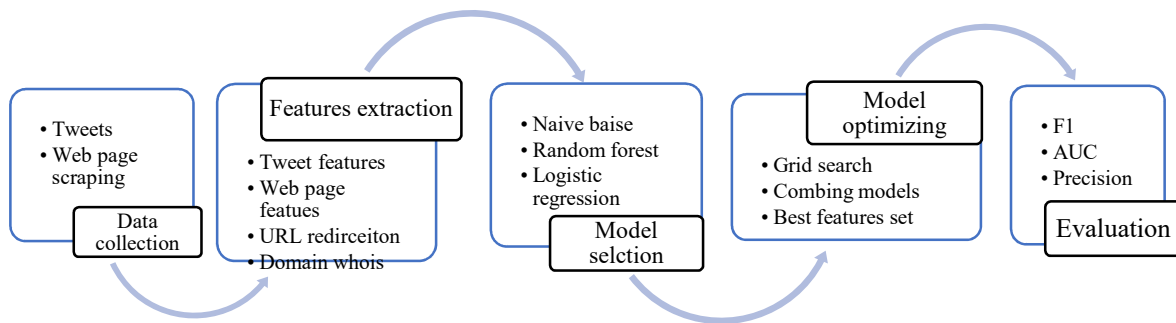


Figure 3.11 Research methodology key phases

Several algorithms were used for building models. The model selection stage here needed to decide which model would be deployed finally. However, before deploying the model, further optimising methods needed to be applied, such as feature selection and model tuning. As the main flaws that machine learning models can fall into are overfitting and underfitting, further evaluation and validation processes were required to ensure that the model will perform in a similar way with future unseen data.

Most pre-processing methods explored in this chapter are feature extraction procedures that aim to extract useful features to be deployed later in machine learning models. The data collecting and pre-processing stages are the most important part of building a machine learning system, as building a reliable dataset that is free from noise and misclassified examples will have a high impact on the model performance.

Features used in building a model can be classified into two generic types based on the effort and pre-processing required for extraction and transformed into a machine learning acceptable format. Novel features have been discussed in this chapter such as content that is generated through the way to reach the final landing page. In general, in this research, the aim is to explore features that require high effort from spammers to disguise and hide them, as it seems to be easy for legacy features such as tweet text and metadata to be manipulated by spammers. Furthermore, the general idea is to take advantage of using the combined features derived from several sources.

The next chapter provides a comparison of some of the supervised machine learning algorithms. It also outlines the tuning and feature selection methods applied to enhance the algorithm with the best performance.

CHAPTER 4

Using supervised machine learning algorithms to detect suspicious URLs in Twitter

4.1 Introduction

There are several supervised machine learning algorithms that focus on a binary classification problem (for example in this work spam/normal); preliminary experiments were conducted to find the best algorithm (using default hyper-parameters) that suits the classification problem and dataset in this research. All preliminary experiments in this chapter are based on DS1, which is the first version dataset collected (see chapter 3, section 3.5). To determine the best algorithms for the spam classification, the performance was evaluated using several metrics and other criteria that have been discussed in chapter 3 were explored. The algorithms investigated are the top common algorithms used in related work, which are DT, RF, LR, NB, and k-NN. Finding the best algorithm was based on the easiness to understand, performance, tuning, and resources required in building and training the model. After finding the best algorithm, more time was spent on investigating what the key tuning hyper-parameters and feature reduction methods are. The experiments in this chapter can be summarised as follows:

1. Several models were built using different algorithms with a minimum tuning process (Sklearn default hyper-parameters were used). Then the models were evaluated using several metrics to select the model that performed the best with minimal tuning required, as the aim of this experiment is only to come up with a classification algorithm family that suits the problem and dataset in this research.
2. The selected model from step 1 was further investigated regarding its main hyper-parameters and how they can play the main role in its performance. Selected hyper-parameters will not be generalised for any other dataset, but the workflow can be applied to any other machine learning solution.

3. The final experiment was conducted on the effectiveness of using several feature selection methods such as filters and wrappers on the performance of the model by reducing the dimension of the dataset to see how the model behaved against a changing feature set size.

The following sections describe in detail how the previous experiments were implemented.

4.2 Model selection

It is common in the context of spam detection for researchers to compare several machine learning algorithms to select the best algorithm for their collected dataset [86] [58] [137]. Researchers have mainly used supervised machine learning algorithms such as NB, k-NN, RF, and LR for spam classification. For the comparison, the same DS1 dataset was used (see chapter 3, section 3.4). In terms of model performance on classification of spam and non-spam URLs associated with tweets, the top four common algorithms reported in the previous chapter (see section 3.4) were used. All algorithms were implemented using scikit-learn³¹, which is an open source machine learning library in Python.

In the preliminary experiment, four classifiers trained and tested using the same set of 36 features and the same training and testing datasets. To evaluate the models, the ground truth dataset was randomly divided into a 75% training and 25% testing set. The Scikit-learn train/test split function generated the training and testing samples of the whole dataset. Ten samples were generated using different random seeds, with

³¹ <http://scikit-learn.org/stable/>

remaining the class percentage as possible by activating stratify option while generating the samples using the Sklearn library. The four classifiers (RF, LR, k-NN and NB) were trained and tested. The models chosen are examples of different types of algorithms. Although the author is aware of the new implementations of ensemble learning algorithms such as XGBoost and CatBoost, the aim in this stage of the study is to determine what type of method suits the nature of the data and the classification problem in this research. The Scikit-learn default parameter values were used for all four algorithms. To give the overall performance of each metric, the researcher averaged the performance over ten experiments.

Table 4.1 Overall performance (average of ten experiments) using one classifier for all attributes

<i>Model</i>	<i>AUC</i>	<i>F1</i>	<i>Precision</i>	<i>Recall</i>
RF	0.92	0.92	0.96	0.89
LR	0.67	0.63	0.67	0.60
NB	0.58	0.62	0.51	0.78
k-NN (k=2)	0.80	0.78	0.79	0.76

The results shown above confirm that RF had the best performance. This aligns with most of the studies in the literature [91][138], which conclude that RF gives higher classification performance than the other supervised machine learning algorithms based on their dataset. Although one of RF's main advantages is that it does not require a fine-tuning process for its parameters [139] to achieve high performance, choosing the right values of hyperparameters avoids overfitting/underfitting. If there is no limit in tree depth the resulting very long and complex trees may over fit the data.

In terms of performance, the results show that the RF classifier with scikit-learn default parameters (10 trees, undefined max depth and leaf size) reached 92 per cent in

the AUC and F1 performance metrics. Although other models might perform better if the researchers reconsidered tuning the models' hyper-parameters, as a first pilot study, the researchers in this study just wanted to find the best models by only using the default parameters that come with Scikit-learn. Therefore, in terms of performance, it is clear that RF outperformed all the other trained supervised learning algorithms. Nevertheless, in general, the pilot study and most of the previous studies agree that RF is one of the top algorithms that could help in this type of data and classification problem.

4.3 Model performance enhancement

One of the main stages in building a practical machine learning model is model evaluation and optimisation, i.e. tuning its hyper-parameters with the aim of improving its performance while at the same time avoiding overfitting. Tuning a model is a trade-off, as increasing the complexity of the model could make the model behave very well on the ground truth training dataset but very poorly on validation of the unseen dataset, which causes the overfitting problem. However, oversimplifying the model would also give a poor result, especially when used for a complex problem like spam detection where many features may be involved in decision-making. Simple models might miss learned important insights and features of the trained dataset, so the model would show poor prediction performance. Therefore, the researchers needed to build a well-balanced model configuration, which could be achieved by hyper-parameter tuning and feature selection and reduction.

To enhance the model, an experiment was conducted to reduce the number of features used in building the model. The fewer features used, the less complex the model will be, which leads to fewer chances to produce overfitted models. For feature selection, two types of method were used, which are filtering and wrapper, each of which might

give a similar but not identical feature set due to their different ways of ranking features. The following sub-sections will show in detail what procedures were needed to find the best model configuration and eliminate features that would enhance the model other than complexity.

4.3.1 Model enhancement through parameter tuning

Despite the extensive use of RF classification for detection of spam/malicious content in OSNs, there is a lack of detailed information about how this method is used in terms of parameter settings [86]. Not giving clear information regarding the hyper-parameters used in building/training models could limit the reproducibility of the reported results and validation. This practice also makes it difficult to understand the impact of parameters on the performance of RF classification applications for OSN spam detection. Although RF does not require high effort in fine-tuning, setting improper RF hyper-parameters could lead to an over-fitted/under-fitted model, which could give low detection performance when tested on new data. To explain the process of tuning the highest-ranked algorithm in the first experiment, which is RF, first, the author needed to identify the top hyper-parameters that have a high impact on the RF model. Based on the literature review, the most important performance-affecting RF hyper-parameters [73][74] are tree number, maximum depth and minimum leaf size, which are mostly specified as the depth of trees could grow (stopping criteria) (as shown in Table 4.2).

Table 4.2 Random forest main hyper-parameters

<i>Parameter</i>	<i>Description</i>
Tree number	Number of trees in building the RF classifier
Max depth	The maximum depth that the tree can grow
Min leaf size	The minimum number of leaves a branch can have

As there is only two free hyper-parameters, maximum depth and minimum leaf size, trees are numbered logically, and the more trees in RF the better. To find the best parameter values for the model, the Scikit-learn grid search method was used. Using this approach, the parameters could be varied based on a range of pre-specified values. All options for all parameters cannot be considered, especially those that can be infinite, such as tree number and maximum depth. Therefore, in the following tuning experiments, high, medium and low numbers were assigned for such infinite hyper-parameters.

In this experiment the following hyper-parameters of RF classification considered: the number of trees, the maximum depth of trees, and the minimum size of leaf nodes (i.e. of the data subset associated with such nodes). The number of data features (i.e. data dimensionality) was kept unchanged, features selection/reduction will be discussed later in section 4.4. For each parameter setting, 20 experiments were run with randomly generated samples (stratified by target) for training and test datasets. To analyse the impact of parameter settings on the classification performance, the average performance and the standard deviation of the performance metric across the 20 experiments were calculated. The performance of the classifiers was measured in terms of recall, precision, and F-measure. The performance results were compared using the t-test to determine whether the difference in mean performance is statistically significant at the significance level of $p=0.05$. For the comparison of standard deviations (in fact, variances), the F-test was used with the significance level at $p=0.05$.

The number of trees

Generally, it is expected that the greater the number of trees, the better the performance [140]. This was confirmed by the results (see Figure 4.1). The results also show that the standard deviation of the performance values decreases as the number of trees increases. These results are valid for all settings of maximum tree depth and leaf node size.

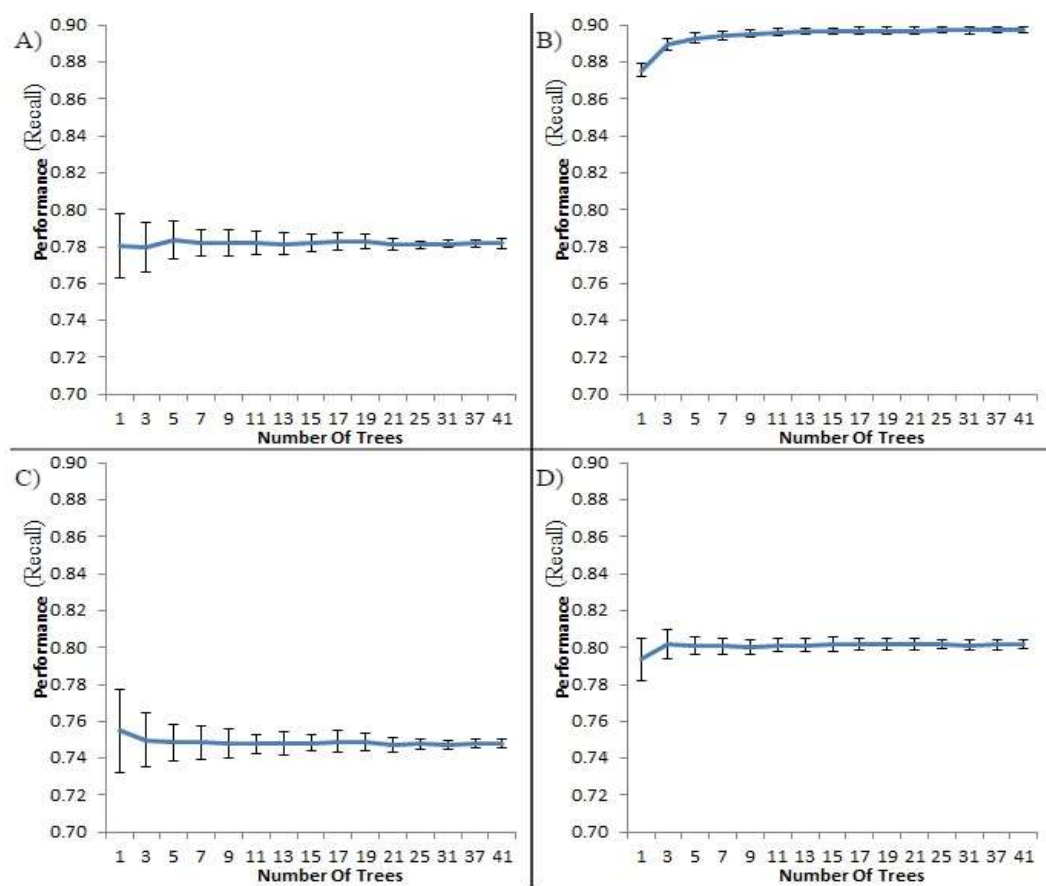


Figure 4.1 The effect of the number of trees parameter on the performance of the spam classification

The leaf size and the maximum depth of trees are different constant values in the four panels: A) max depth = 10, leaf size = 10; B) max depth = 44, leaf size = 10; C) max depth = 10, leaf size = 300; D) max depth = 44, leaf size = 300.

The effect of the increase in the number of trees is most prominent in terms of mean performance for random forests with large maximum tree depth. For small maximum tree depth, adding more trees to the random forest improves the performance in terms of reducing the standard deviation, although there is not much improvement in terms of mean performance. In all considered cases, the mean performance does not improve statistically significantly beyond nine trees in the random forest.

The standard deviation of the performance improves more for random forests with smaller maximum tree depth than for those with larger maximum tree depth. The improvement is statistically significant (at $p = 0.05$) up to 25 trees and becomes insignificant beyond that.

Maximum tree depth

Maximum tree depth is one of the variables that determine the complexity of the RF classifier. Trees can be built without any depth limit; however, in general, it is recommended to control the tree depth to avoid overfitting [141]. Here, the effect of varying maximum tree depth was analysed, while considering a range of fixed combinations of the number of trees and minimum leaf size.

The results show that the maximum depth of trees has a major effect on classification performance in the context of the spam detection problem in this research for all considered combinations of number of trees and minimum leaf size values (see Figure 4.2). It was found that the mean performance of the classifiers improves with the increase in the maximum depth of the trees. This improvement is statistically significant (at $p = 0.05$) up to maximum depth 16 for random forests with large minimum leaf size and up to maximum depth 24 for random forests with small maximum leaf size. It was

also found that the standard deviation of performances gets smaller as the maximum depth is increased and that this effect is the strongest for RF classifiers with large minimum leaf size and few trees.

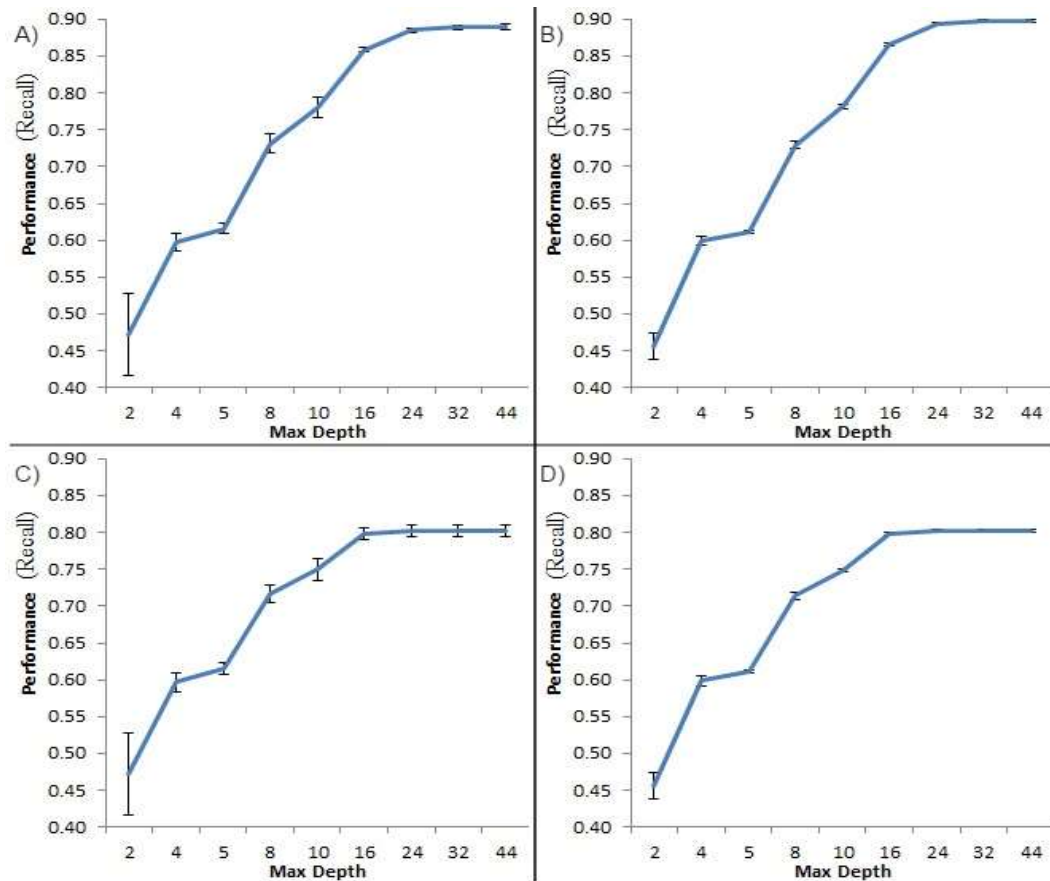


Figure 4.2 The effect of the tree max depth parameter on the performance of the spam classification

The leaf size and the number of trees are different constant values in the four panels: A) number of trees = 3, leaf size = 10; B) number of trees = 41, leaf size = 10; C) number of trees = 3, leaf size = 300; D) number of trees = 41, leaf size = 300.

These results show that setting the maximum tree depth too low leads to low classification performance irrespective of the minimum leaf size and the number of trees (see Figure 4.2 for maximum tree depth below 8 in all four panels). Setting the maximum tree depth high may not lead to trees with that depth due to the limit on the minimum

leaf size; however, setting low leaf size and high maximum depth could lead to overly deep and complex trees that could show high performance on the training dataset but poorly when tested on unseen examples.

Minimum leaf size

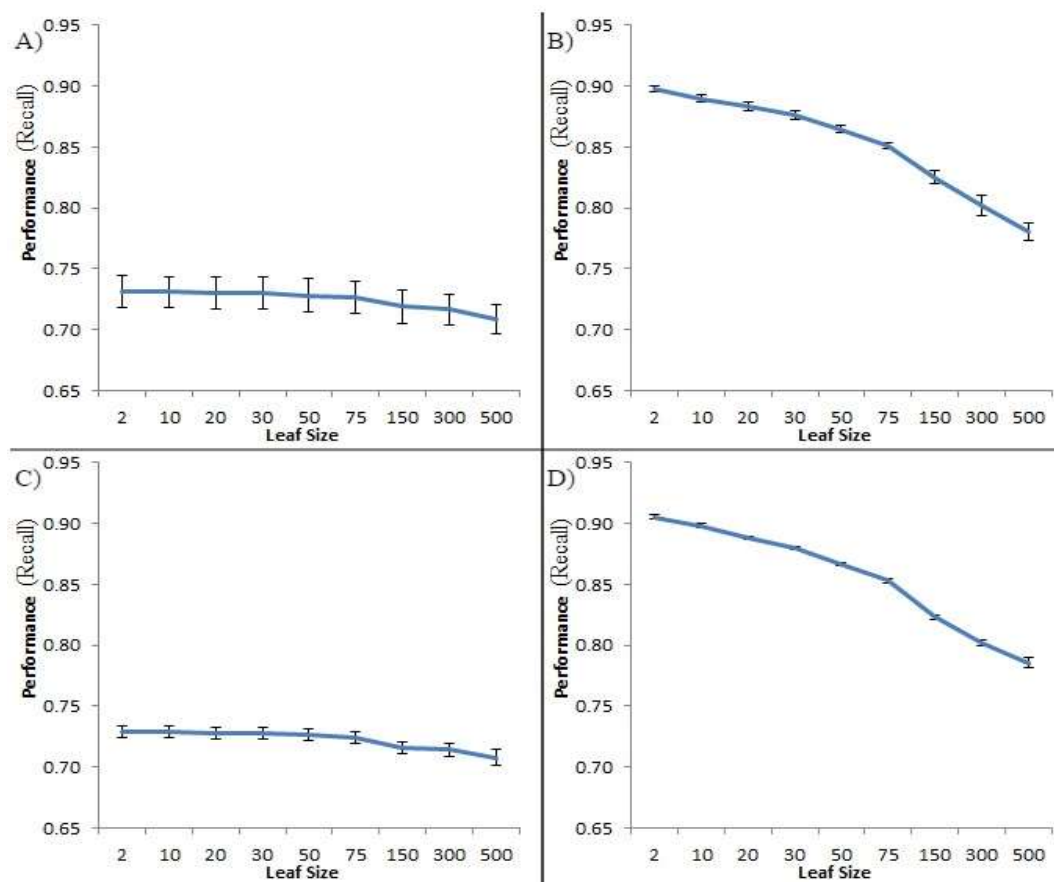


Figure 4.3 The effect of the leaf size parameter on the performance of the spam classification

The maximum tree depth and the number of trees are different constant values in the four panels: A) number of trees = 3, max depth = 8; B) number of trees = 3, max depth = 44; C) number of trees = 41, max depth = 8; D) number of trees = 41, max depth = 44.

The minimum leaf size controls the complexity of the decision trees by setting a size limit for the data subsets associated with leaf nodes, consequently preventing the adding of further decision nodes to the tree after the nodes reach this limit. The effect of

changing the minimum leaf node size have been investigated while keeping the number of trees constant and the maximum tree depth for RF classifiers applied to the spam detection task.

The results show that the increase in the minimum leaf size reduces the performance of the classifier in all cases (see Figure 4.3). This effect is much more pronounced in the case of classifiers with high maximum tree depth than in the case of classifiers with low maximum tree depth. This because the similar effect of both parameter, short trees have large leaf size and vice versa. The effect is similar for different numbers of trees in the classifiers, the only difference being that for a large number of trees, the standard deviation of performance is lower than for a small number of trees.

For RF classifiers with low maximum tree depth, the minimum leaf size has a statistically significant effect on the performance if it is larger than 30 (number of trees) or 50 (few trees). This indicates that in these cases, the limited depth of the trees implies the limited performance of the classifiers for smaller minimum leaf sizes. Conversely, for classifiers with high maximum tree depth, the effect of the leaf size is statistically significant for all values of this (i.e. larger minimum leaf size implies significantly reduced performance).

4.3.2 Experiment summary

The experiments' analysis shows that the parameters of the RF classifiers, number of trees, maximum tree depth and minimum leaf size, are important determinants of the performance of these classifiers. In the context of the spam detection task in this research, the classifiers that were built perform very well if the number of trees is sufficiently large, the maximum tree depth is sufficiently high, and the minimum leaf

size is sufficiently low. However, the generalising power of the classifier could be affected by how deep trees are and minimum leaf size. Although the number of trees is always better to support performance and make the classifier less over-fitted to the dataset, performance enhancement could indicate the feasibility of increasing trees in random forest.

The results show that the number of trees has a relatively small impact and that beyond certain number of trees, however more trees could stabilise the performance of random forest classifier. The minimum leaf size has more effect, especially for classifiers with high maximum tree depth, for which even small changes in the minimum leaf size have a significant impact on the performance. Finally, the maximum tree depth has a significant effect on the performance for low values of this parameter and the effect diminishes below significance for depth values above 16 or 24 for small and big minimum leaf size respectively.

This implies that the number of trees and maximum tree depth should be set to moderate values to achieve good performance without an excessive computational burden and be less prone to overfitting. A minimum leaf size that is too small combined with a maximum tree depth that is excessively large is likely to lead to overfitting (note that the overfitting is because of the trees and not because of the forest arrangement of the trees [79], [142]). Therefore, controlling the minimum leaf size is important, and again it should be set to a moderate value to avoid overfitting and excessive unnecessary computation. The experiment results show that the performance changed dramatically by changing certain hyper-parameters, for example, the depth size in one tree model or number of trees hyper-parameter in a random forest model. Therefore, clearly stating the

hyper-parameters of the machine learning model would help other researchers to replicate models.

4.3.3 Hyper-parameter tuning and overfitting

Overfitting is a potential problem for decision tree learning [142] and consequently for RF classifiers as well (note that the number of trees does not cause overfitting by itself [79]). Dealing with this is important, since excessively good results generated by overfitting decision tree solutions of classification problems are misleading. In particular, this is an important issue in the context of OSN spam classification because of the popularity of RF classification in this application domain and the potential impact of incorrect classification of social media messages.

The results confirm the expectation that imposing a limit on the maximum depth of the decision trees and on the minimum size of the data subsets associated with leaf nodes of the trees reduces the potential for overfitting. The results quantify these limits and the impact of going beyond these limits in the context of the particular dataset of non-spam and spam tweets.

The number of trees in the RF classifier mainly impacts the standard deviation of the classification results. The number of trees also has an impact on the amount of time required to train the classifier (the required time is proportional to the number of trees). This means that at the expense of the computation time, the robustness of the classification results can be improved by adding trees to the random forest. However, the results also show that the gain in reduction of the standard deviation of the classification performance becomes insignificant beyond a certain number of trees.

The work in this study implies that in general, when RF classification is applied to spam detection in OSNs, the impact of maximum tree depth, leaf node size and

number of trees should be assessed to determine the sufficient values of these so that overfitting is avoided and performance gains are realised. This also means that the results of such applications should be reported with sufficient metadata about the application, including the number of trees, maximum tree depth, minimum leaf size, and any other parameters that have an impact on the performance of the RF classifier for which results are reported.

Naturally, the results in this study are limited in terms of specific values that were found for RF classification parameters regarding the tweet dataset that was used. However, the main conclusions from the analysis illustrate the importance of the determination and reporting of RF parameters. Moreover, this process is not limited to this classification problem but is also valid for any classification application of this method.

4.4 Model enhancement by feature selection

Selecting the top feature set is one of the important parts of building a machine learning model, as reducing features reduces the complexity of the model and the overfitting risk. However, the method used to remove features needs to be studied since the model is primarily built on good features, so removing them could lead to a rapid decrease in the model's performance. Evaluating features, which is also known as the feature importance score, is an essential process to understand the dataset that researchers rely on to build models. Moreover, it enables the researcher to distinguish between good features and irrelevant features. Eliminating redundant and noisy features could cause performance improvement [143]. There are several existing methods to perform feature selection, such as the wrapper and filter methods [144]. The feature selection procedure was conducted in two stages. In the first stage, each feature in the

original feature set was evaluated by applying a number of available feature selection metrics. Second, top k features were chosen which should have the highest discriminative power and ignored features that have negative impact on the model's performance [67].

The wrapper method concept is based on model performance, and every chosen subset of features is used to build a classifier and evaluate its performance until the optimal subset is found with the lowest error rate. For a high-dimensional dataset, the wrapper method could be a costly and time-consuming method; however, the wrapper method is one of the highly efficient methods as its feature evaluation relies directly on the classifier performance. It has been shown [145] that the wrapper method achieves higher classification accuracy than the filter method. Despite the high computational resources required, it is recommended that the wrapper method is applied for such classification problems when the number of features is within the computing capability. As the aim of this research is to increase the detection performance, and the number of features is within the computational resources the system used, this method was considered for further investigations.

The MDA [146] is determined by ranking features based on the decrease in performance value after removing features one at a time. Essential features should show a negative impact when they are removed. Conversely, less important features should have no significant negative impact when removed. However, as mentioned earlier, some features acting as noise could have a negative impact on the model. Removing such features might improve the performance of the model.

As feature correlation is an issue in building a machine learning model, multicollinearity happens when some features have a correlation with each other.

Features could have a different shape or relation such as correlated and causal, redundant or noisy. Therefore, it is important to have domain knowledge to understand features and their relations. Features could rationally understand the logic correlation such as account age and tweet number, so a newly created Twitter account should have a low tweet number. On the other hand, it could be more complicated to discover some correlations without statistical tests and feature selection algorithms. Feature selection would also help to get rid of such correlated features that have a negative impact on models. When a decision tree-based model is built, if there are correlated features when the tree is split on one of the correlated features, the importance of the other one will be reduced as its affect has already been applied by the correlated feature. However, the importance of RF features will give unreliable feature importance as both correlated features will be giving a somewhat relevant feature importance, which is due to the sampling.

The method used in feature selection in this thesis could handle the issue of multicollinearity by removing features and measuring whether removing a feature whether would have a negative, positive or even no effect. Therefore, removing correlated features may not show any effect on the model or it could enhance the model performance. The negative impact of this method is that the model might lose semi-correlated features that needed to be pre-processed, for example, combining them in a feature dimension method such as principal component analysis.

The filter method uses importance measurement methods to assess the information content of features and possibly their correlation with the target classification. Unlike the wrapper method, the filter method does not rely on classifier performance to rank features' importance, making its application much faster. Table 4.3

shows the features' importance ranking based on three methods, information gain, Gini Index (filter methods) and MDA, which is a wrapper method. It worth mentioning that the results below are from the first dataset (DS1) collected during the study.

Table 4.3 Ranking of features based on information gain, Gini index and mean decrease average

#	<i>Feature</i>	<i>Info. Gain</i>	<i>Gini Index</i>	<i>MDA</i>
1	Domain age (WHOIS)	1	2	1
2	Number of digits in link (URL)	2	1	2
3	Number of external links (web page)	7	6	5
4	Ratio of age to number of tweets (Twitter)	3	3	8
5	Link letters (URL)	4	5	11
6	Number of links (web page)	8	7	7
7	Number of images (web page)	11	9	4
8	Number of dots in link	13	13	6
9	Ratio of words to external links (web page)	6	10	12
10	Number of input forms (web page)	12	12	3
11	Number of words (web page)	10	11	10
12	Link length (URL)	5	8	13
13	User statuses count (Twitter)	9	4	15
14	Number of link signs (web page)	14	14	14
15	Number of ad blocked links (web page)	20	18	9
16	User friends count (Twitter)	15	15	16
17	Account age (Twitter)	17	16	17
18	User followers count (Twitter)	16	17	22
19	User favourites count (Twitter)	18	19	18
20	Number of hashtags (Twitter)	21	21	20
21	User listed count (Twitter)	19	20	24
22	Does link contain 'www' ³² ?	25	24	19

³² <http://www.yes-www.org/why-use-www/>

#	<i>Feature</i>	<i>Info. Gain</i>	<i>Gini Index</i>	<i>MDA</i>
23	Does web page have password input? (web page)	27	25	21
24	Link letters (URL)	22	23	27
25	Does tweet have media? (Twitter)	28	27	23
26	Number of mentions (Twitter)	23	22	26
27	User name length (Twitter)	24	26	30
28	Is https protocol used in URL? (URL)	26	28	25
29	User name digits (Twitter)	29	29	31
30	Is tweet is a reply tweet? (Twitter)	33	31	28
31	Number of URLs (Twitter)	32	33	29
32	User name signs (Twitter)	30	30	33
33	Is user geo-enabled? (Twitter)	31	32	32
34	Default profile image (Twitter)	34	34	34
35	Is user account verified? (Twitter)	35	35	35
36	Is user account protected? (Twitter)	36	36	36

The features' importance is varied; each method ranks features' importance somewhat differently, although there is general agreement on the top and bottom, which are the best and worst features. In this stage, the author aimed to select the top k features that give the best classification performance. To conduct feature selection, first, the lowest-ranked features were eliminated from the three ranking lists that were produced by three different evaluating techniques. Therefore, at each number of features, an RF classifier was built based on the new feature set, then compared it to the original performance achieved by using the feature set with all the original 36 features. The stopping criterion was whenever the performance was statistically less than the performance of the first classifier the researchers built using all features, which was 0.89 in the recall. Figure 4.4 shows the performance of classifiers against the number of

features used. Each time the RF classification model was evaluated, 10-kfold cross-validation method was used to evaluate classifiers based on the recall metric.

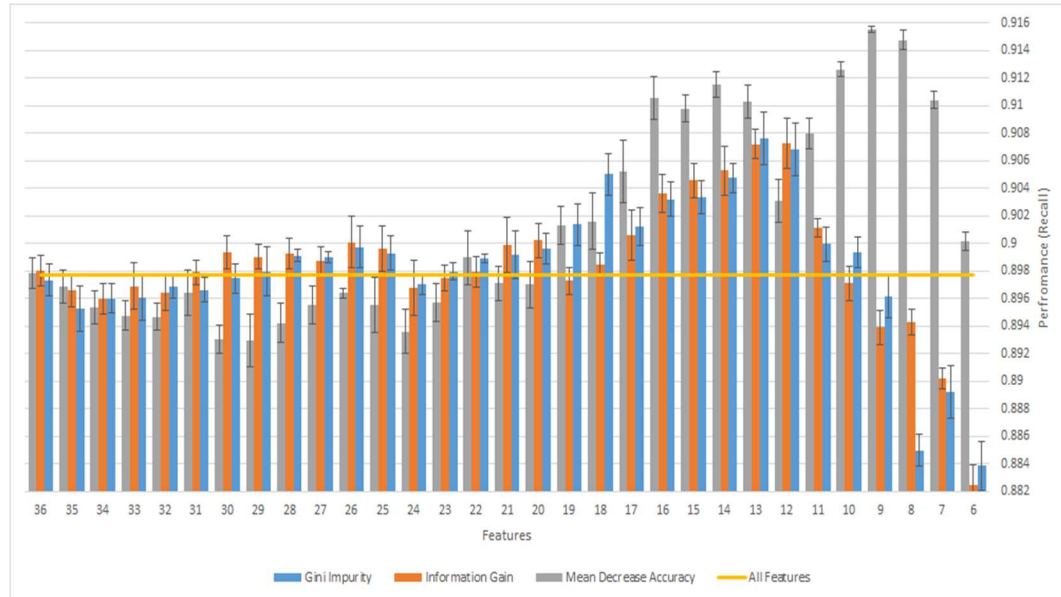


Figure 4.4 RF classification performance based on the selected features. The vertical axis is the performance in Recall and the horizontal axis shows number of features as it is decreasing from left to right. The yellow horizontal line is the model performance using all the features.

In Figure 4.4, the horizontal axis represents the number of top features used to build the classifier, and the vertical axis represents the performance in recall. To assess the impact of features on the classification performance, features removed one by one from the three ranking lists, starting from the original 36 features. The performance of the classifier that was built using the original feature set (with 36 features) is shown as the horizontal line fixed with the performance at 0.897 in recall in Figure 4.4. This was used as a benchmark performance to assess the extent of improvement or degradation in classification performance caused by elimination of features. Figure 4.4 does not show classifiers' performance for less than six features, as the performance drops considerably with further reduction of the number of top features. The classifier performance

improved as the lower-ranking features were removed. The filter methods reached their peak performance (0.908) for 13 features for the Gini impurity features ranking list and for 12 features (0.907) for the information gain features ranking list. Conversely, the MDA-based elimination of features reached its best classification performance (0.916) for nine features. All subsets of features that were generated from different methods show improved classifier performance compared to the model built using all the features.

4.5 Enhance model performance by adding more training data

The more data used for training, the better the model will perform. However, in some cases, the resulting performance enhancement is small compared to the difficulty of acquiring accurate labelled data. Figure 4.5 illustrates the model's behaviour according to the size of the data used in training. It shows that models reach a point where the chance of overfitting becomes less likely. However, the model improvement reaches a point where adding more data does not significantly change the performance. This led to further investigations being conducted to improve the performance further.

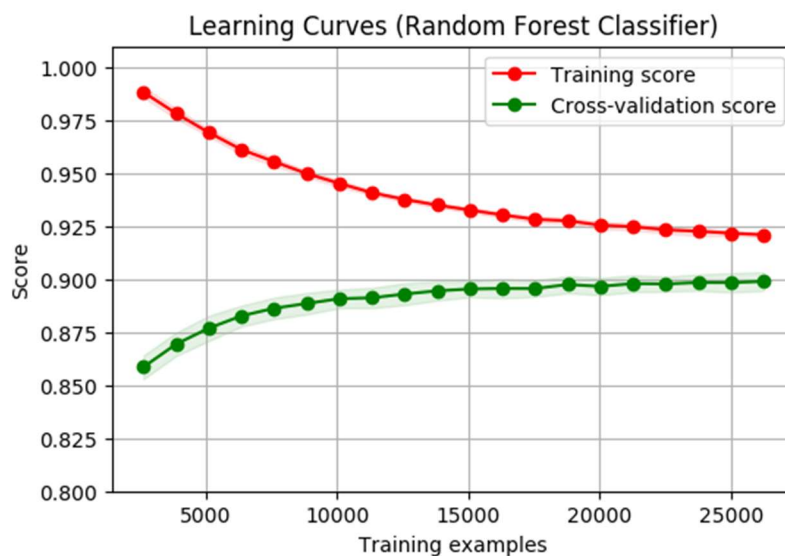


Figure 4.5 RF incremental learning curve according to DS1 random forest performance according to the size of training data used

From 20,000 training data points onwards, the cross-validation performance did not change significantly, which means that adding more training data points would not lead to significant performance improvement. Therefore, better discriminative features needed to be found instead of continuing to collect/extract the same features used in DS1. That is why DS2 was collected, which has a higher number of features and was labelled with an extra manual validation stage to develop an accurate ground truth dataset. DS2 was used in all the experiments described in the next chapter so that a better model could be built using a dataset with higher dimensions. The next chapter will discuss the use of more complex models that was investigated. More advanced and complex models can be achieved by adding more discriminative features to the dataset used [146].

4.6 Conclusion

In this preliminary experiment, the author first investigated several supervised machine learning models to find which type of algorithm gives good results without going further into tuning it. Further time and effort was invested to optimise the chosen algorithm (RF) and come up with the best hyper-parameters for the dataset, the highest performing model using the smallest number of features and the smallest structural parameters (tree number, max tree depth and maximum leaf size), to find the least complex but high-performing classifier.

As RF showed the highest performance among the compared algorithms, the researchers went further to determine the main hyper-parameters that limit the model's performance and complexity. Experiments show that the tree number hyper-parameter in RF gives more stable results; however, it could reach a point when more trees do not add any significant impact to the performance. However, trees' stopping criteria hyper-

parameters, such as tree depth and minimum sample leaf, have a direct impact on the model complexity, so choosing the optimal values that give good results but are not overfitted to the trained dataset is essential for model evolution. For parameter tuning, the aim was to determine the best setting for the number of trees, the size of leaf nodes and the depth of the trees in the RF classifier. However, these numbers would be suitable only for the training set used.

Then another experiment was conducted to study three feature selection methods that belong to two techniques, which are the filter and wrapper methods. First, the required minimal structural hyper-parameter values of RF were determined. Following this, a model was built using all the features and the feature set was reduced to the minimally required set. The best feature set reduction was achieved using the computationally costly MDA wrapper method; however, relatively similar performance (although statistically significantly lower) and feature set reduction was achieved by using the two chosen filtering methods as well.

In summary, the results show that the process of hyper-parameter tuning is essential and can make a difference in terms of finding the balance of high performance and encountering an overfitting problem. Furthermore, the feature selection process could enhance performance and reduce the resources required for extracting expensive unnecessary features or by reducing the dataset dimension. Focusing on using only important features in building the classification model reduces the model's unnecessary complexity. It is also important to report the parameter values and the details of the feature set optimisation method that was applied to guarantee the reproducibility of the results reported in studies.

CHAPTER 5

More Informative Features and Ensemble Learning Methods Used to Detect Malicious URLs on Twitter

5.1 Introduction

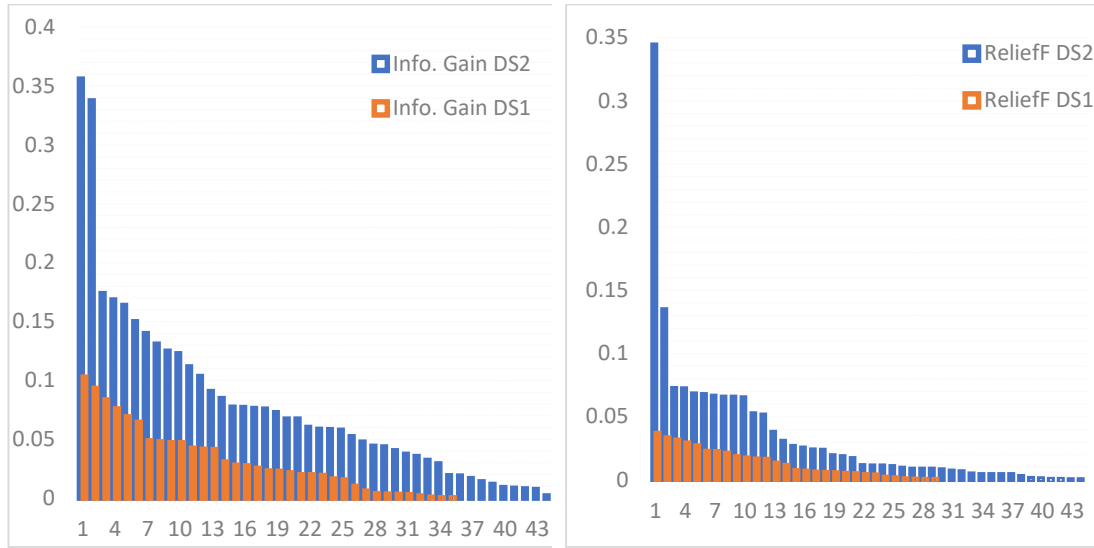
The previous chapter gives a clear understanding of what is required when building a machine learning-based detection method, such as model selection, tuning, and feature selection. As an ensemble-based model (RF) showed better results, further investigations were conducted by studying several ensemble methods using tree-based models. More ensemble learning models are used in this chapter's experiments, such as XGBoost, extra random trees and gradient boosting trees. Moreover, in the second part of these experiments, several ensemble learning models were combined using two methods of combination. Combining models could aid in optimising performance by producing an even better model or/and could contribute to the process of model selection. Therefore, in this chapter the aim is to investigate the possibility of achieving both or at least one of the previous goals.

It was also shown in the previous chapter that the model enhancement reached a point where no further improvement could be achieved even when more data was used in training. Therefore, to overcome this issue, all the experiments applied in this chapter are based on DS2, which contains deeper features extracted from several sources. To obtain further information about the nature of this dataset and what new features were added and what features were eliminated, see section 3.3 in chapter 3.

5.2 New dataset and new features deployed

The previous dataset and model could not get higher than a certain performance regardless of how much more data was presented, which can be solved by building a more advanced model. The model complexity can be adjusted by the hyper-parameters or by adding more features. As building a supervised machine learning classifier requires a highly accurate and reliable ground truth dataset, in this experiment the researchers

used another dataset (DS2), which has higher discriminative power features and of which the labelling process was more robust and precise.



A) Information Gain ranking

B) ReliefF ranking [34]

Figure 5.1 Comparison of DS1 and DS2 features importance based on two feature ranking methods

Figure 5.1 shows a comparison between DS1 and DS2 features importance measured using two different methods information gain and relief. Using both features importance ranking method DS2 have shown it has higher features importance compared to DS1 and also more features. Due to this robustness and more complicated features being extracted, DS2 is smaller (6,000 tweets/URLs) than DS1 (the majority of features come from the social network side). More details about the labelling and the features extraction methods that applied in collecting and building both DS1 and DS2 are outlined in section 3.3.

5.3 Ensemble learning methods

Several learning algorithms have been investigated in the domain of spam content detection on OSNs. Among the most common algorithms used in previous

studies [58][86], the ensemble learning algorithm RF more often outperforms others such as a solo learner (k-NN, DT and NB). In studies such as [86][58][17], algorithms that are based on ensemble techniques such as RF and boosting trees showed better performance than a single model. This is consistent with previous observations which show that ensemble models show better performance than individual models [76].

Table 5.1 Common features and classifiers used in the literature (algorithm with highest performance identified by bold type)

Study	Learning Algorithms	Features
[86]	SVM, k-NN, RF , DT and NB	Twitter meta features (text not used)
[58]	LR, k-NN, RF and NB	URL, Twitter, web page content
[17]	XGBoost , LogitBoost and RF	OSN metadata and behaviours (worked on fake likes)
[147]	RF, C4.5 (DT) and Combined classifiers (built using different datasets)	Statistical features derived from Twitter metadata
[148]	Regularised SVM and Regularised Logistic Regression	Twitter metadata, text similarities
[137]	RF , SVM, k-NN, GBM and XGBoost	Posting pattern

According to the literature review, RF is one of the most common algorithms used in this domain. In terms of performance against other supervised learning algorithms, it is often ranked highest (this also aligns with the researchers' preliminary study). However, several ensemble learning algorithms recently started to compete with RF and gradient boosted trees, which are XGBoost [78], LightGBM, and CatBoost. All new implementations of the gradient boosted tree method have started to attract increased interest and give better accuracy and speed in learning.

Ensemble models consist of several models which could all be built using the same or different types of machine learning algorithms. The classification decision is made based on the ensemble method used, which can use unweighted or weighted voting. Unweighted voting means that there are not priorities for classifiers and all have

the same decision power. Ensemble learning aims to perform better and be less prone to overfitting than a single model [76]. For more details of all the ensemble learning algorithms used in this experiment, see chapter 2, section 2.4.4. Boosting and bagging are both ensemble learning techniques. Bagging uses several base learners trained on different samples, whereas boosting is a single model repeatedly trained on a weight-adjusted dataset. These differences mean that they might develop different learning hypotheses on the same dataset, which in this study suggests that it is potentially possible to integrate the two methods' algorithms into one stacked model, which could result in an even better model that complements each method.

5.4 Experiment methodology

This chapter's experiments aim to find a way to use several models and combine them in a way that can make them work in calibrated methods. Coming up with such a way would avoid the stage of selecting and comparing models, as the main aim of the thesis is to develop a system that can automatically retrain and select models. Other experiments also investigated in more detail more advanced ensemble learning algorithms that belong to boosting and bagging methods. The experiments in this chapter can be summarised as follows:

1. Ensemble (boosting and bagging) and non-ensemble learning algorithms were built using DS2 and evaluated and ranked based on their performance.
2. Models were sub-grouped into three groups based on the range of their performance (all models, top five models, and top three models).

3. All groups were combined using two methods that were equally weighted, the voting and stacking methods, as this experiment will help in finding a way to automate the model selection process by combining them.
4. The combined method was examined to determine which one will be more suitable to take the role of model selection in SuspectRate.

As ensemble learning is a promising approach, in this study, the author has investigated several ensemble methods applied to the problem of malicious URL detection on OSNs. Therefore, in this chapter, the experiments started by comparing the most common and new ensemble learning tree-based models to see which give better results with the dataset and nature of the problem in this research.

Using the DS2 ground truth dataset, several bagging and boosting classifiers were built: RF, extra trees (extremely randomised trees), AdaBoost, and XGBoost. For each model, the author used stratified 10-fold cross-validation, as this is considered an efficient way to prevent the overfitting problem [149]. The experiment was repeated 20 times by varying the stratified 10-fold seed every time.

In this experiment, unified common hyper-parameters were used for all the classification algorithms using decision trees as the base learner. For research reproducibility, the parameters used to build the base classifiers and overall ensemble models are as follows. The number of trees used in RF and extremely randomised trees was 333. The author also used 333 to represent the number of boosting iterations before combining the models' predictions. Regarding the tree's maximum depth, in the boosting model, the number was fixed at 6, but 10 was set in the RF trees and extremely

randomised trees. Trees in RF and extremely randomised trees grew until they reached a leaf with a minimum of 30 leaves. These numbers were chosen after conducting experiments on tuning each classifier's number of hyper-parameters. Generally, boosting methods use weak base classifiers (short trees) for training, unlike RF algorithms, which build full deep tree classifiers [83]. RF, AdaBoost, extra trees and gradient boosting algorithms were implemented using Scikit-learn (an open source machine learning library in Python). However, XGBoost was implemented using an open source library in Python and R [78].

In the second part of the experiment, the aim was to study two ways of combining methods which are better in terms of overall performance and require less human intervention. The first method averaged the individual predictions; the second method used the stacking ensemble method to come up with the final prediction from all the predictions inputted into the meta-learner classifier. The models used were derived from different methods of learning, such as trees, nearest neighbour and logistic regression, and the detection performance could range from high to low classifiers.

5.5 Results and evaluation

To evaluate the classifiers, the commonly used evaluation metrics of ROC-AUC, Logloss, and F-score were used. Table 5.2 provides a brief definition of the evaluation metrics used in this study.

Table 5.2 Definition of metrics

<i>Metric</i>	<i>Brief definition</i>
F1 score (weighted)	The average of recall and precision (weighted by the percentage of each class). Weighted means that each class score is averaged based on its percentage in the test dataset.
Logloss	Represents the certainty of model prediction probability as the best value is 0.
Roc-auc	See sub-section Classification performance on page 97.

Table 5.3 presents the overall performance of all the implemented models. The models with the highest performance are highlighted in bold according to the metric, and those that are underlined came second. Ensemble learning classification algorithms came first in terms of performance.

Table 5.3 Results of models using stratified 10-fold cross-validation method

<i>Model</i>	<i>Logloss</i>	<i>F1</i>	<i>ROC-AUC</i>
XGBoost	0.2258	0.9080	0.9056
RF Classifier	<u>0.2906</u>	<u>0.9055</u>	<u>0.9037</u>
Gradient Boosting Classifier	0.3221	0.9050	0.9028
Extra Trees Classifier	0.4654	0.8617	0.8541
AdaBoost Classifier	0.6771	0.8916	0.8896
Gaussian NB	0.7914	0.7468	0.7372
Logistic Regression Classifier	0.5086	0.7507	0.7411
K-Neighbours Classifier	0.6143	0.6597	0.6513

All ensemble classifiers achieved at least 85 per cent in ROC-AUC (with Extra Trees showing the lowest performance). Generally, non-ensemble models show lowest-performing classifiers which are Gaussian NB, K-Neighbours Classifier, and Logistic Regression Classifier. The Scikit-learn implementation of gradient boosting trees performed slightly worse than bagging RF; however, XGBoost was ranked first when using the same trees numbers with an increase in classification performance to 0.904. The majority of all the metrics used to compare the performance of models showed that the top three performing algorithms were XGBoost, RF, and Gradient Boosting Classifier.

The results of the second part of the experiment are presented in Figure 5.2, showing several sets of models built and combined using averaging and stacking

methods. Three sets of models were studied to determine which combination method suits the dataset and nature of the problem in this research. Table 5.4 shows the three model sets used in this part of the experiment. Figure 5.2 shows that the averaging top selected models work better than the stacking method. However, when there is diversity in a model's performance, the averaging method performance declines; this happened in the group of the best five and the set with all the models.

Table 5.4 Model sets used in combination methods

<i>Model Set</i>	<i>Models Involved</i>
All Models	All models in Table 5.3
Top 3 Models	XGBoost, Gradient Boosting trees and RF
Top 5 Models	XGBoost, Extra Trees, RF, Gradient Boosting and AdaBoost

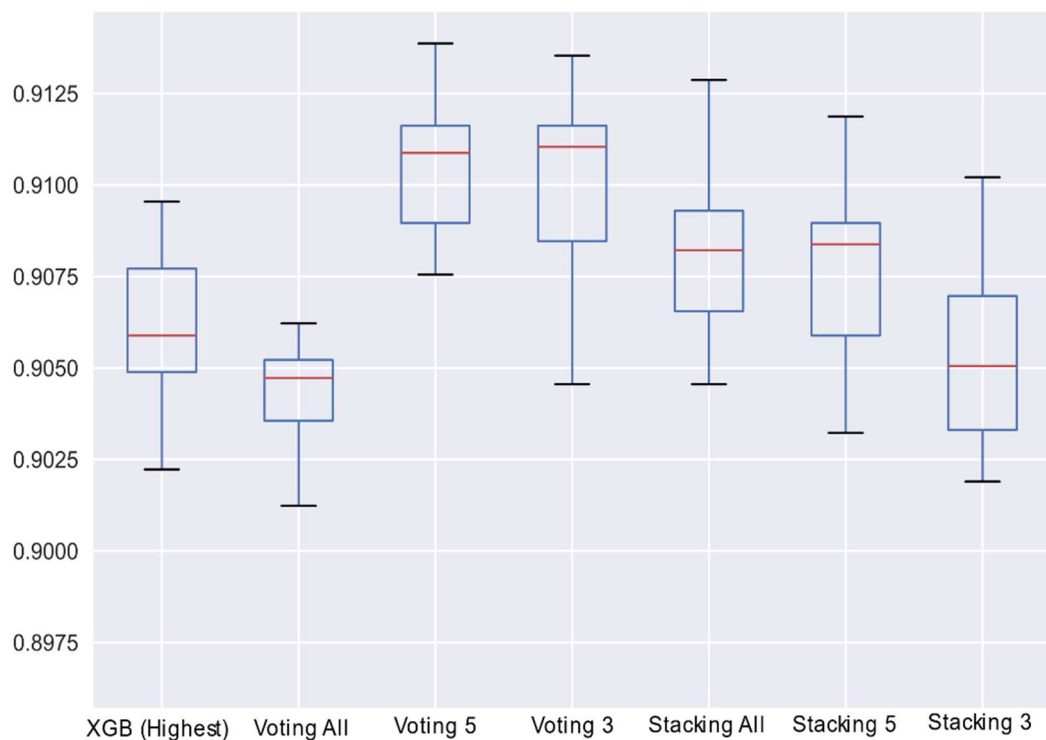


Figure 5.2 Models' performance according to F1 metric model on the left XGB is the highest model compared with varied combined models groups

Although combining methods such as the voting method (soft) and stacking did indicate a statistically significant improvement, there were relatively small advantages compared to the best singular model, which is XGBoost. However, the advantages of these methods will be clearly shown when the model is used later in the SuspectRate spam detection systems to automate the process of model selection. A framework was being developed that supported automating the process of model selection by stacking several heterogeneous or homogeneous models. However, preferably, stacked models have a different learning methodology or are even built using different hyper-parameter settings. The author aims to use the model selection/combining method that can exploit each model's strengths and is not significantly affected by the model's weaknesses.

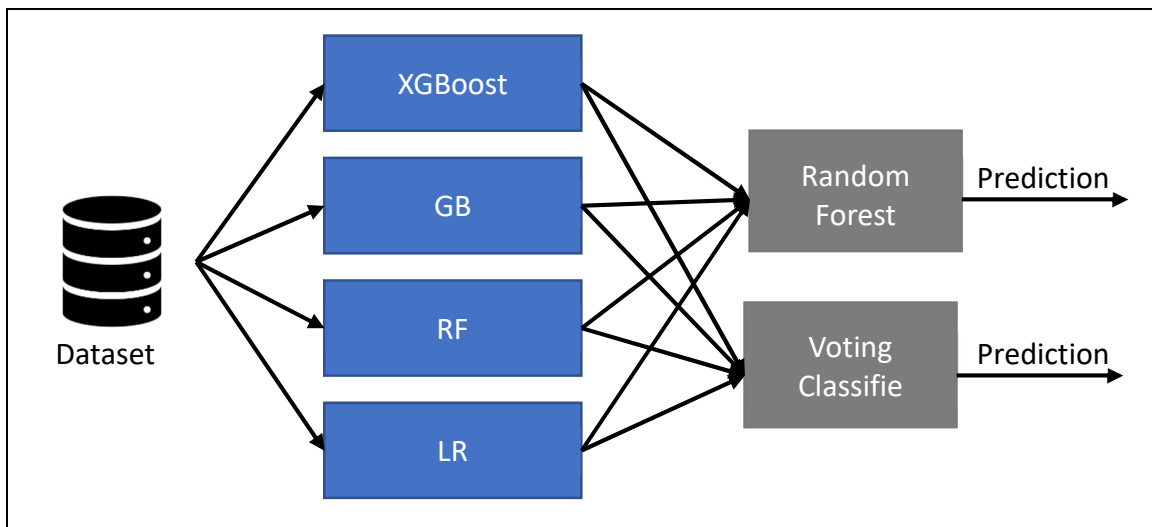


Figure 5.3 Methods used for combining predictions of several machine learning models

The stacking method shows better performance in integrating different models in terms of method of learning and performance. At the same time, the stacking method does not decline rapidly (as averaging methods do) when weak classifiers are stacked up with the models. This is due to stacking split datasets into subsamples and using

evaluated features (first stage classifier predictions) for each sample, so it is unlikely that one classifier is always the best for each random sample. Therefore, weak classifiers will not be chosen unless they show some advantages on the subsample of the dataset, where they could be chosen and help the overall prediction performance. Voting classifiers which work on averaging the predictions output of the input classifiers achieve better and more stable performance when input is limited on top classifiers only. However, the process of model selection is necessary to filter out all the weak classifiers before averaging the top ones to achieve better performance.

In general, combining models using bagging, boosting or stacking helps in achieving better results. Regarding how to choose among different models using different ensemble methods, in this study, two methods could be used to fully and partially automate the process of model selection among several available models. Figure 5.4 shows the top three models' and top two combined models performance using F1, accuracy and roc-auc and it show two possible ways to combined heterogeneous models voting and stacking. Top three models voting mainly used if the main priority is selecting the highest performance, an extra step is needed for ranking and evaluating the models. In the case of stacking model, when there is no need to evaluate the models, the predictions should be given to another machine learning model to make the decision of selecting the best combination of models. As the aim is to build an automated spam detection system, the stacking method will be used that shows more robustness in dealing with heterogeneous models even when weak classifiers exist in the stacked classifiers. Therefore, the next chapter will describe the developed framework 'SuspectRate' and explain how the stacking method helped in automating the process of model selection.

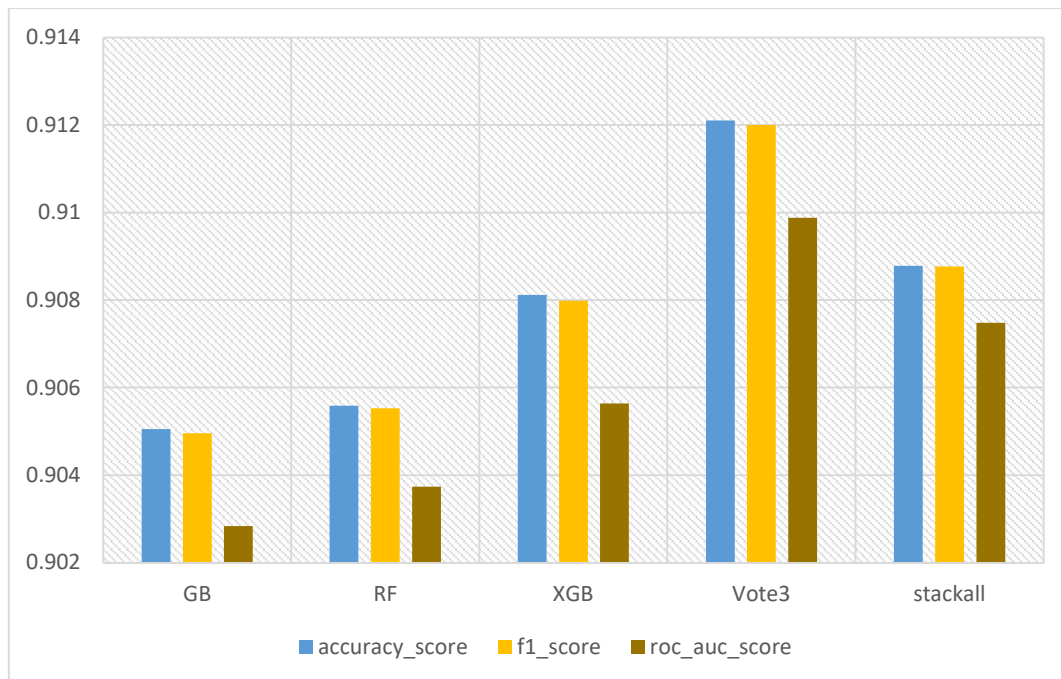


Figure 5.4 Top models vs top combined model

5.6 More data, better performance

To gain a full understanding of the classifiers used, the author compared the experiment and its behaviour while increasing the training data size. In the previous chapter, the researchers did test the impact of adding more data for training. Although DS1 was larger, it reached a point when adding more data showed no significant improvement.

Figure 5.5 shows the top three models (RF, XGBoost and Gradient boosting trees) and top voting and stacking models performance behaviours on the vertical axis over the horizontal axis, which represents the used percentage of the training dataset. Figure 5.5 shows the top three ensemble models' learning curves; each curve plots the F1 value against the number of training examples. The points of the curves represent the F1 average of stratified 10-fold cross-validation. Figure 5.5 shows the models' performance using the F1 metric, where it is evident that at the first training split, most

models had very similar performance, and then the stacking model and combined top three classifiers model started to outperform in almost all the portions of the training dataset.

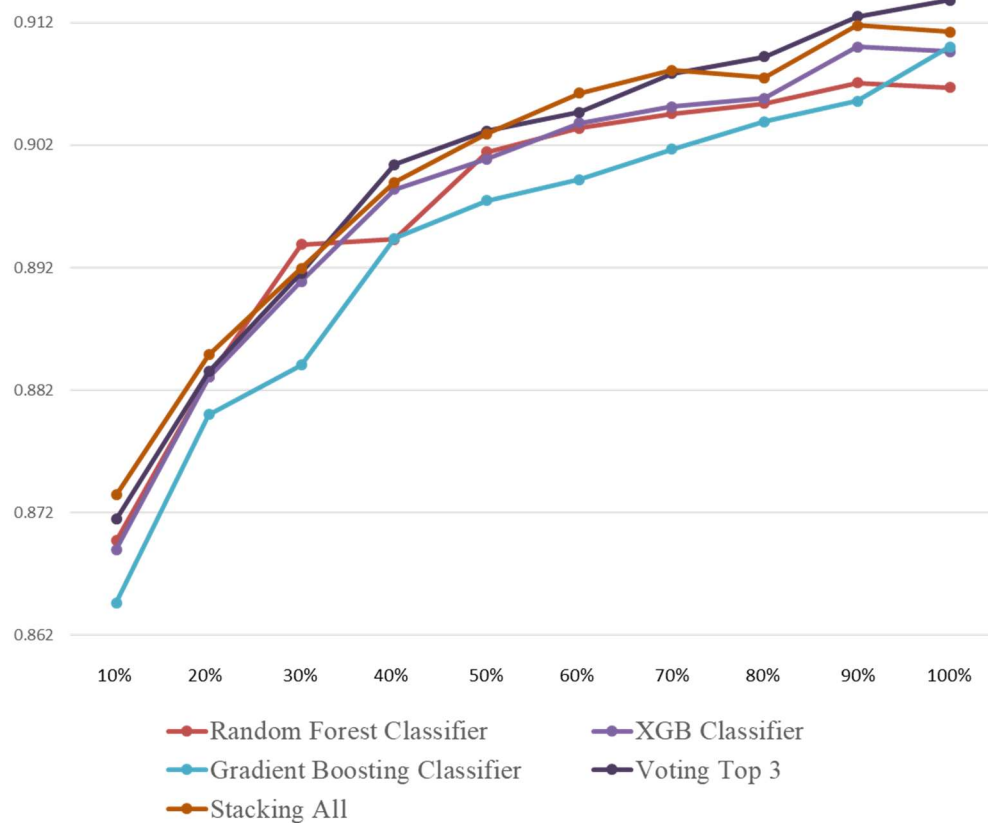


Figure 5.5 Learning curves for ensemble learning models using the F1 metric.

The figure shows the classifiers' performance behaviour as more data is used for training.

The learning curves of the algorithms on the dataset have similar trends among the algorithms. As Figure 5.5 shows, F1 performances increases in an almost constant value when the number of training instances is increasing. When these learners learned more than a quarter of the dataset, the combined model tended to outperform other

models. The results obtained indicate that in general, the models' performance is enhanced when more training instances are provided. Furthermore, compared to the previous experiment in section 4.5, better results were achieved with a small amount of data but more informative features and using more advanced ensemble models.

5.7 Conclusion

Several experiments are reported in this chapter, which starts by comparing the two datasets built during this research and how they differ in terms of the number of features and their importance. One of the main methods to build even more complex and advanced models is using more and better discriminative power features or by using more training dataset.

Then further ensemble classification algorithms that rely on different learning mechanism such as boosting, bagging and stacking were used. Boosting and bagging algorithms such as XGBoost, gradient boosting trees and RF gave the models with the highest performance. For stacking, the researchers aimed to find a model with even better performance that combined several boosting and bagging models and performed as one calibrated model. The aim was to find a method to abstract the process of selecting the best model.

As there are many different types of spam content, such as advertisements, pornography and fake software or malware distribution, no single classifier can be suitable for all these types of spam. Different algorithms respond differently to dataset characteristics, such as class imbalance, noise, and outliers. For example, boosting gives better results than bagging when trained on purer datasets, whereas bagging algorithms are better for handling noisy (collection and features extraction errors) datasets because of the bootstrapping process [150], [151]. This is the advantage of combining several

ensemble classifiers and using the strengths and weaknesses of each to support the others in the decision-making process [152]. Thus, the researchers can contribute to the process of building smart detection systems that integrate more than one machine learning algorithm based model and able to retrain classifiers periodically. Each classification algorithm could look at the data from a distinct perspective, so a system could be used for different types of spam.

The majority of previous studies that have compared several algorithms for spam detection in social networks have shown that ensemble learning algorithms achieve the best performance. In this chapter, two ways of combining models were tested, which showed that it is possible to achieve better and more stable performance by combining several heterogeneous models (boosting, bagging and non-ensemble-based models) than rely on solo model. Besides the enhancement archived by combining models, used combining methods can be used to automate the model selection process.

CHAPTER 6

‘SuspectRate’ – a Spam Detection System

6.1 Introduction

Social networks accept content of users that has an attached URL leading to more details or referring to an external page outside the social network platform. Those URLs lead to external content which can pose various types of threat to the users, ranging from compromising their connected devices to exposing them to low-quality content. Although the blacklists used in all OSNs are useful to eliminate the content that has already been discovered and listed, the real challenge is when the OSNs' system receives a new spam URL/domain with no history. Spam URLs/domains with no history get passed by the blacklists filter and are distributed to thousands of users in real time. The time gap between content with suspicious content attached being distributed through the network and being detected and listed on a blacklist is what security centres and researchers are trying to narrow.

Consequently, a system designed and developed to reduce this gap and provide decision support for a security administrator to update blacklists in a shorter time. The system's goal is to be near to the real time, reliable and able to generalise to detect newly suspicious content with high accuracy. SuspectRate, is a tweet (with URL) analysis system that developed by the author. SuspectRate can crawl URLs and extract features to check them against pre-trained machine learning models.

A decision in the developed system is based on features that were previously studied and proved their effectiveness in chapters four and five. The features are based on analysing the lexical and URL structure. In addition to features that represent the behaviours of the web pages, JavaScript events that were triggered by a page loading/unloading event were used. Moreover, web page behaviours were used against events that the developed system generated to test the web page, such as mouse click

and page closing event. As there has been a rise in popup windows leading users to different pages, this could be the real threat that the attacker wanted to lead the OSN users to.

In this system, when the researchers refer to suspicious URLs or content, this means any content that could range from low-quality untruthful content to harmful malware or virus content. Adult content (pornography sites) is considered as spam content based on Twitter terms. Therefore, the goal of this system is to provide a suspicious rating for every attached URL in streamed content in OSNs.

6.2 Design goals

As discussed in section 2.6, one of the drawbacks in current machine learning-based solutions is that they are built and trained on a constant ground truth dataset that could represent the spam methods and tricks at that time, but spammers are continually developing new tricks that can bypass those systems. Therefore, there is a need to build a framework that makes the proposed machine learning models adaptable and does not lose its effectiveness over time. The researchers plan to provide OSN platforms with a tool to detect suspicious content and remove it. The design goals can be summarised in these main points:

1. To automate the process of deploying a machine learning model to detect suspicious content on Twitter
2. To have the ability to be adaptive to the new techniques and tricks that spammers use to bypass systems
3. To have the ability to auto retrain internal machine learning models
4. To automate the process of model selection
5. To provide highly accurate and reliable classification

6. To have the ability to be updateable in terms of data use for training and models.

Currently, the systems work as software as a service (SaaS), where users are required to send links of their timeline and keywords they want to follow or collect and the system will assess and send tweets' IDs back to the users with an associated number that represents the tweet suspect rate. The practical value of the system at its current development status is to be used for collection and labelling tool for those who rely on social network as a data source. For example, researchers or analysis companies who rely on Twitter need to spend high effort on collecting and cleaning tweets and its attached data before performing further analysis. As twitter could contain high percentage of spam and low-quality content that have no real impact on studying real human interaction on social media. For thus, this system can be useful for collecting data from twitter and each tweet will be evaluated from zero as genuine to one as suspicious, so researchers will have the ability to assign the best threshold percentage to accept or discarded tweets. Saving researchers time in collecting and cleaning and make them more focussing on building better models would enhance research productivity.

But one of the future work plans is make an API access to this system, so several solutions can be built on the top of this service. For example, web browsers extension that automatically send post/tweets URLs to SuspectRate via API requests and give early warning to users if it has high suspicious rate.

6.3 System flow and structure

The system's main components can be divided into external components that users connect to and internal components where the system does all the training and

model evaluation and selection. To help users to connect to the system, a web interface was built that has forms for submitting a task and checking the status of the submitted task. Internally, there are several components, tweet collection, feature extraction, and data preparation and classification, which are presented in Figure 6.1. All the components are shown in Figure 6.1, and the components and flow of the system are described in detail in this section.

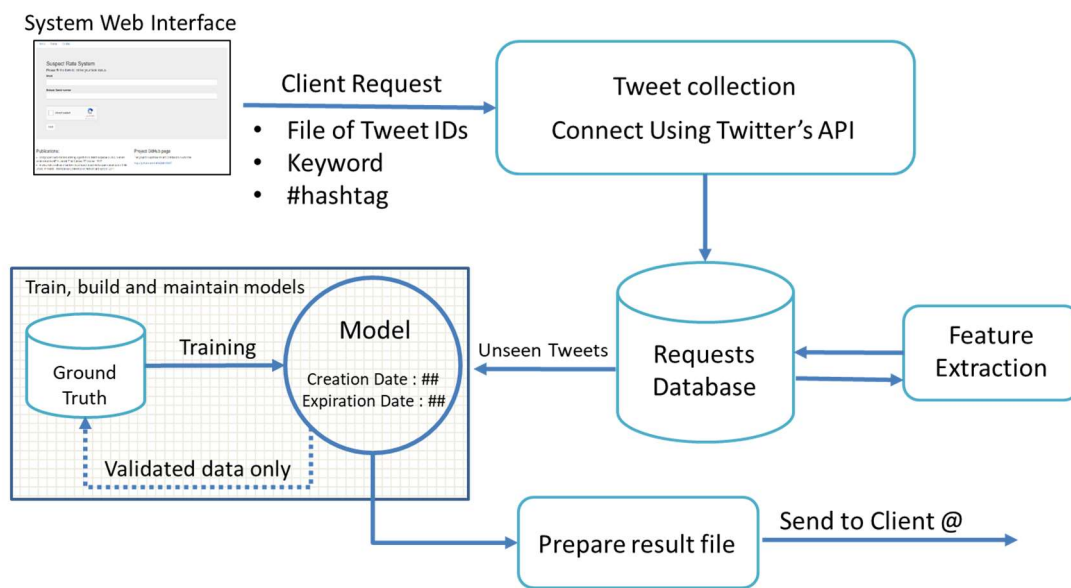


Figure 6.1 System's main components
flow starts from system web interface and ends by sending email to the client,
small grey area represents the internal retraining process that conducted
periodically

All components are developed from scratch starting from building a web interface and uploading datasets to the system host and then importing data into a database. Collecting data depends on the type of source, which could be direct tweet IDs or monitoring specific keywords or hashtags. Then feature extraction components (see Figure 6.2) are developed as each one requires specific and customised methods to obtain the required data. Moreover, system models are maintained and an automatic

retraining schedule is internally programmed to be run daily. Finally, the results are prepared and the applicant is notified that the results are ready to download.

Task input (using Web GUI): The system's web interface is the main port to use the researchers' prototype system. Users can submit their task through simple forms that require name, email and file, keyword or hashtag. Users can submit tweet IDs for the researchers to carry out the collection and sequences process or if users want to collect data from the tweet stream that contains specific keywords or hashtags. After a user submits a task to the system, they will receive a confirmation that contains details of the task ID and the expected finishing time or date. They will also be referred to the status page where they can check the task completion percentage.

Tweet collection, feature extraction and data preparation: Connecting to Twitter to retrieve a task tweet is the first thing the system does, and then all retrieved tweets are stored in a database (called requests DB). The feature extraction component starts once a new tweet is added to the table. This component contains several subcomponents that focus on extracting groups of features from various sources. Figure 6.2 shows in detail the inner process applied to each request to get into this component, starting with getting the final landing page from the unshortened tweet's URL (if shorted link) and ending with the process of combining all the features and transforming them into an accepted machine learning format.

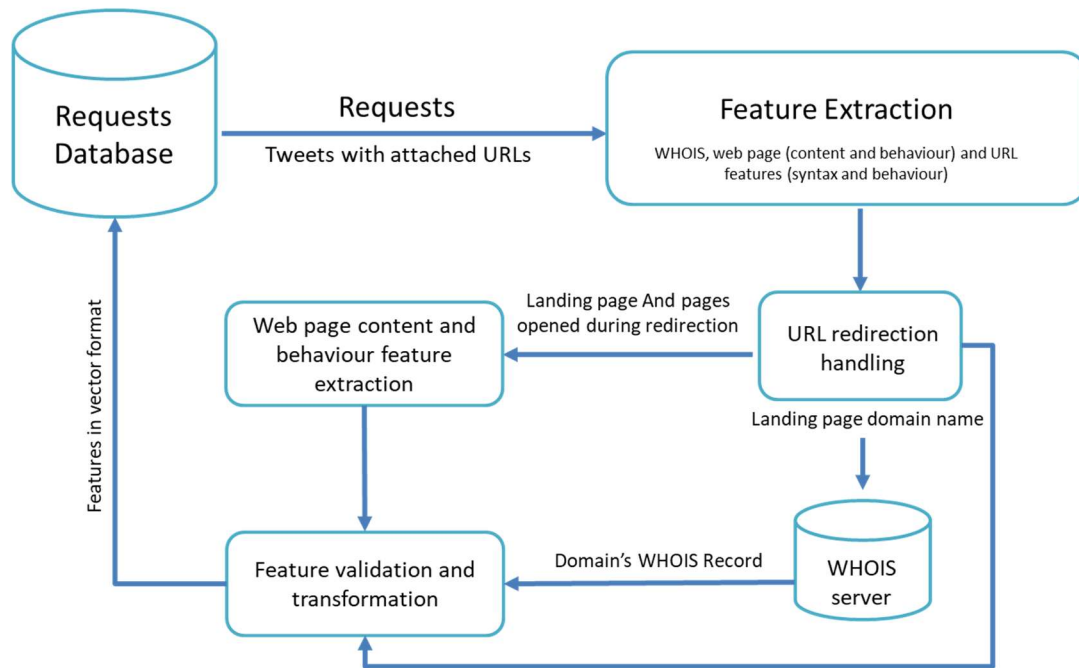


Figure 6.2 Feature extraction internal processes
extracting feature process start by retrieving URL from the database and ends
back with all extracted features to the database.

The URL redirection feature shows the number of redirection URLs included before reaching the landing page. The pages opened during this URL redirection process were stored. Once the researchers reach the landing page, they start crawling the page using Selenium WebDriver (Google Chrome and Firefox). The landing page domain name is then passed to WHOIS feature extraction, which does domain WHOIS information requests and parses response data to obtain valuable information about domain ages and registrar information. In the final process of the feature extraction component, the system's feature validation and transformation component carry out the final data validation to check whether all the features that needed to be used in the detection model exist and have been transformed into an accepted machine learning format, which is called the machine learning domain features vector. The process of

checking features' formats is done automatically without any human intervention. Features should follow certain criteria and validate them before being inserted back into the database.

Finally, all the features and data collected are returned to the requests database; however, this time it is flagged as ready to be used against the system detection model. For further information about the features used in this system and the implementation and techniques involved, see chapter 3, section 3.3.1.

Classification stage: This stage contains pre-trained machine learning models which will be used to give the suspicion probability for each tweet (with URLs attached). In the back end of the system, there is already a specific component of which the main goal is to conduct model training and model selection. The next section will give more details about how the models were built and trained. In general, the task in this stage is to retrieve flagged requests and apply them to the deployed model or models depending on the method used. Requests should have a similar number of features and format to the ground truth dataset used in training the models. The classification output is a probability number that represents how suspicious the content is. The value of suspicion level ranges from 0 to 1, 0 being not suspicious at all and 1 being very suspicious of being a spam tweet.

6.4 Building and training models

After evaluating several machine learning classification algorithms in previous chapters, choosing the best for their classification was a challenge. Explored algorithms ranged from simple linear classification algorithms such as logistic regression to more complicated ensemble learning-based models. Based on preliminary studies (chapter 6),

ensemble-based models showed high performance in the researchers' ground truth dataset. Moreover, as the designed system aimed to detect suspicious content, which can come in different forms such as pornography, illegal advertising, scam and phishing, no one model can be perfect for all these types of suspicious/spam content. Therefore, in this detection system, the aim was to automate the problem of choosing the best classifier by using stacking and voting methods which have been used and tested on real a dataset in section 5.5 in the previous chapter. The ensemble stacking method, where several heterogeneous machine learning algorithms are combined into one model, was used.

Table 6.1 Models used in first version of SuspectRate system

Models	Ranking
XGBoost	1
RF Classifier	2
Gradient Boosting Classifier	3
Extra Trees Classifier	4
AdaBoost Classifier	5
Gaussian NB	6
Logistic Regression Classifier	7
K-NN Classifier	8

The models used in this first version of the developed system are shown in Table 6.1. All the models were derived from supervised method machine learning algorithms. They will all be involved in making the classification, and the model's vote will be weighted based on the performance of each model.

6.5 Deploying and maintaining models

One of the important characteristics of the designed system is maintaining and retraining models. Models are built differently according to the training dataset, so when new labelled data points are added, the system will retrain the models. The researchers aim to make the system retrain itself every time a certain amount of data is added. Since the system's decision is made using several models' decisions, the researchers needed to produce a new combined model to be used for classification. The system will not be affected by the retraining process since it will continue to use the old model until the new model is ready for deployment. Each model used will be stored with a date name to keep backup models in case any model is trained using corrupted or mislabelled data.

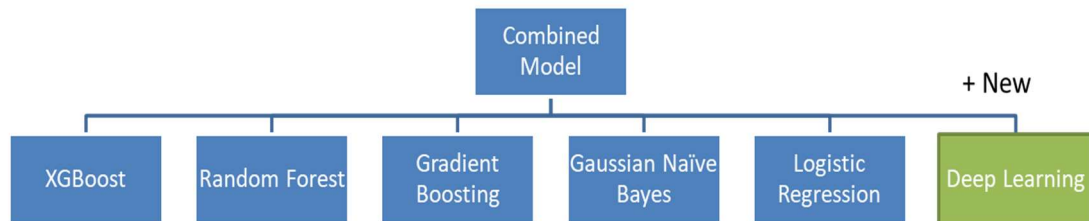


Figure 6.3 System's ability to add new models
green box shows a new deep learning model added to the classifiers stack

Furthermore, one of the key features of the designed system is its ability to add new machine learning models in the future. For example, the researchers aim to add deep learning and deep forest models to be used with stacked models. Therefore, the system will be able to be enhanced in two ways, either by adding new training data or by adding new models that have proven to be suitable for the researchers' system classification problem.

6.6 System data input

The input stage is when users want to contact the system to send tweets that need to be assessed by the system machine learning model. Currently, the system has one input channel which is via web form, but in the future feature, it is planned to make the system connected through an HTTP API. Figure 6.5 shows the developed web interface that enables the user to send a file that contains a list of tweet IDs. The requirements are simple name and email, and then the file can be uploaded. The email field is compulsory as the system will send an automatic email to the user when the file has been fully analysed. The submitted file is simply a file that contains tweet IDs in each row.

Tweet ID1
Tweet ID2
...
Tweet ID3
Tweet ID4

Figure 6.4 Input file
one column csv file, each row contains a tweet ID

After the tweets are receive, the system filters them and accept only tweets that have at least one external URL and store it in the system database (MongoDB). As described in section 6.3, tweet collection, web scraping and feature extraction are components that always in checking requests DB for any new requests. Once a new tweet is received, this component starts by obtaining the tweet using the tweet API and then extracts all the tweet information and attached URLs.

Home Status Contact

Suspect Rate System


Spam filtering is an important service for all variants of electronic communications. Spammers on social media are difficult to detect in time. We aim to use a combination of machine learning methods and optimised selection of features to develop a novel spam filtering system for social media.

Name

Email (make sure to submit the correct email because results link will be send to this email)

Image file

No file chosen

☐ I'm not a robot 

Publications:

- Using supervised machine learning algorithms to detect suspicious URLs in online social networks MF Al-Janabi, E de Quincey, PE Andras - 2017
- A systematic analysis of random forest based social media spam classification M Al-Janabi, P Andras - International Conference on Network and System, 2017

Project GitHub page

This project is opensource and contribution is welcome

<https://github.com/mohfadhii/NSS17>

**Figure 6.5 Web form for uploading the dataset
three fields required name, email and dataset file**

6.7 Feature extraction

This stage is a highly time-consuming stage due to the multiple sources for features that need to be extracted. Several techniques are required to obtain a tweet's URL and open the landing page while recording all extra pages and popup windows that show during reaching the landing page. All web page features are recorded starting with taking a screenshot for each page and then reading the WHOIS information of the URL domain name. SuspectRate needs features derived from Twitter, web page content/behaviours, and domain WHOIS information. The most straightforward features are those that derived from Twitter. Twitter provides data in JSON format with keys that are understandable and easy to access. Some of these features, such as number of followers or friends, come in a numeric format which can be used without any conversions.

Web page content and behaviour features can be one of the complicated features, and the designed system took on average 20 to 30 seconds to extract them for each URL (this period varied based on internet speed and resources). Due to the need to crawl the URLs in a way that makes the bot get into the intended landing page, the majority of tweets' attached URLs are shortened. Therefore, a Python code needed to be built that handles a redirection (JavaScript and http redirections) web page opener that mimics real user behaviours of having a real web browser and follows the URL redirection. Moreover, during this web scraping process, the system is stating whether any popup windows or Java events occur. Events can be automatically triggered, timer's events or event based on reaction to user actions such as mouse click or even movement.

To open a web page automatically, the author used Selenium web browsers. Selenium enabled the researchers to automate the opening of URLs in real web browsers such as Firefox and Google Chrome. Furthermore, the researchers could obtain screenshots of the web pages, running time, content and even pages' behaviours, such as windows or messages that appeared.

6.8 Decision-making and output presentation

After all the tweets' features are extracted and presented in vector format, they are presented to a pre-trained model to obtain the suspicion probability. The probability ranges from 0 to 1, with 0 indicating non-spam and 1 indicating spam. Although some studies prefer to give a direct decision of spam or non-spam, the author in this study preferred to return a probability and enable users to decide what threshold is appropriate for the problem. There is always a trade-off issue between recall and precision, as some problems could accept false positive more than false negative and vice versa.

Consequently, the researchers leave the threshold tweaking to the user; however, the default value is 0.5.

Tweet ID1, 0.9
Tweet ID2, 0.11
...
Tweet ID3, 0.44
Tweet ID4, 0.96

Figure 6.6 Output file
two columns csv file, first contains tweet id then suspicious rate

Finally, tweets and their suspicion probability are arranged in an output file, and then the system automatically sends an email to the user telling them that their file been fully analysed, and the results are ready. The user could also check the status of their requested task using the system status page by providing their email and dataset's submission serial number. The system then shows a page that indicates the current percentage of completion.


Home Status Contact

Suspect Rate System

Please fill the form to retrieve your task status:

Email

Dataset Serial number

☐ I'm not a robot 

Publications:

- Using supervised machine learning algorithms to detect suspicious URLs in online social networks MF Al-Janabi, E de Quincey, PE Andras - 2017
- A systematic analysis of random forest based social media spam classification M Al-Janabi, P Andras - International Conference on Network and System, 2017

Project GitHub page

This project is opensource and contribution is welcome

<https://github.com/mohdadhil/NSS17>

Figure 6.7 Web form to get the status of user order
two fields required (email and serial number) to retrieve order status

6.9 Implementation details

System built using python3 with up-to-date open source libraries and tools so that system can be open source and compatible with all the different platforms. The system components are all built from scratch starting with the web interface and system dataset importing and extracting features. Several text processing and parsing stages are required to extract entities from the web page source code and domain WHOIS record. Furthermore, one of the major parts of the system redirection handling is required to understand what type of redirection method is used so that proper method is used. Moreover, handling popup windows is an essential feature in this system as there is a percentage of spam content shown during the process of redetection to the landing page and not in the landing page itself.

The system speed depends on the number of web pages opened and redirection handled though the feature selection process. Some pages send users to endless windows or dialogs, which annoys users and makes it difficult for them to deal with it. Therefore, the system is pre-programmed to deal with such web pages in case it puts the system in an endless loop of redirection or popup windows. A time and a number of redirections threshold are specified so that the system can exit such cases. In general, the time the system required in its current state is 20–30 seconds on average to extract and predict new tweet/URLs imported into the system. The system detection performance has already been evaluated in chapter 5, although the detection performance could be decrease/increase depending on the maintaining of the system by adding new verified labelled data for the retraining process.

SuspectRate is hosted and built on a Windows server (a virtual private server provided by Keele University³³). The system was developed using Python and many other libraries. The list below contains the essential technologies, open source tools and libraries used to build this system:

- Twitter API – used to establish the communication between the designed system and Twitter to obtain a content stream
- Python 3 – the language used to program the system
- Flask Python web framework – used to design the labelling tool and URL checking page
- MongoDB – the NoSQL database used to store tweets and the extracted features
- Windows server – the system used to host the designed system
- Selenium WebDriver – used to drive a browser natively
- Scikit-learn – a machine learning library used to build most models
- XGBoost – a library originates by a research project at University of Washington³⁴ that gave an API presentation of the extreme boosting trees algorithm.
- Server to host the system (system can work on any Windows/Linux server)
- High-speed internet to speed up to process of web page scraping.
- High traffic network bandwidth to give the ability to open multiple web pages at the same time.

Since the designed system was built using Python and all the system dependencies are Python libraries, the system is theoretically platform-independent.

³³ <https://www.keele.ac.uk/>

³⁴ <http://dmlc.cs.washington.edu/xgboost.html>

However, the researchers have only tried it on a Windows environment. Further details about dependent libraries and tool versions are provided in APPENDIX B.

6.10 Conclusion

In the social networks, the spam detection task requires a fast response to the new emerging techniques and tricks that hackers use. Building a machine learning model is a step not a goal, as maintaining the model performance by retraining the model using an updated dataset is an essential process to keep the system reliable. Moreover, the feature set needs to be updated and re-evaluated periodically to cover more spam activities and content that current features cannot distinguish. Just like in internet security and the spam detection domain, there are no constant robust features, as features that used to be highly discriminative can become less effective if spammers change their methods or content. To continue to maintain the machine learning model, the researchers need to ensure that it is built on reliable and validated feature sets to achieve high-quality performance. To help researchers, companies and even normal users to collect data from Twitter, the researchers who conducted this study offer them a service to assess collected tweets to help them include or exclude tweets.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

The extent of suspicious links in OSNs poses a high risk for a huge number of users. The attackers lure social network users to click their links by using trending, sensitive subjects, controversies in society or sexual material. The crux of the problem of why OSNs cannot detect these URLs when the content is submitted is that OSNs use blacklists as the first validation stage. Since blacklists are capable of detecting only pre-known malicious URLs and most of the links spammers use are either new or shortened, blacklists will be easily avoided by spammers using these newly created URLs. Consequently, there is a crucial need to use systems that are smart enough to detect even new URLs by finding a pattern from previous detected URLs. In this thesis, a machine learning algorithm was used to build a model for spam/suspicious content detection by training the model on pre-known suspicious/normal URLs and content.

Although some common supervised learning algorithms are used in building spam detection models, generally, ensemble learning algorithms show the best performance. However, the challenge in relevant studies is not only the model selection, but also what features that they have extracted and the method of labelling their training dataset. Finding highly discriminative attributes is difficult, as data comes from several sources and in several types, such as text, images, numbers, and behaviours. Here, the role of researchers involves finding the best combination of features to build a machine learning model that is capable of discovering as much spam content as possible.

In this work, the researcher has collected properties from several sources and of various types. The features extracted range from lightweight to heavyweight features. In addition to the lightweight features that the researchers could easily obtain from Twitter, they used texts of tweets, web pages, and information properties of the domain

WHOIS record. Using this combination of features shows up to 0.9055 per cent F1 score using RF and 0.9080 per cent F1 score using XGBoost.

The features used in this study were chosen based on research on current spamming activities and the features that are pre-known from the preliminary studies. Moreover, the feature set used changed during the study, as in DS1 the features were focused more on social network information (lightweight features), whereas in DS2 more powerful features were introduced that were mainly derived from URL behaviour and the domain WHOIS record. The need for a more accurate dataset with better features was identified when the researchers tried to achieve better performance by adding more training data. Therefore, examining the classification problem and improving the model accuracy should be done from all aspects (tuning, feature selection and data revision). Tuning the model in favour of model complexity and increasing the accuracy could lead to an overfitted model. Furthermore, evaluating the features used is essential, since even highly sophisticated models need a reliable feature set to achieve reliable results.

In this research, the author aimed to build a sustainable spam detection system that can maintain its performance even when encountering new spamming. Machine learning-based models in the field of spam detection are not permanent solutions due to the never-ending war between spammers and security researchers. As the researchers used the characteristics employed in previous studies, they found that many of the features lost their value and their discriminatory power. In this domain, researchers do not deal with the naturally derived data, but with the data created by spammers which is modified and created to fool OSN detection systems and even users. Therefore, the developed system can perform model retraining and feature evaluation and selection

periodically. Moreover, the system will be regularly trained on new data that is added to the training database. Since machine learning models are being developed and enhanced rapidly, the system is capable of being connected with a new model in the future without the need for any core changing or building.

Finally, the process of selecting a model from several available models is a challenge for researchers, where the comparison of the model requires more than one performance factor. Therefore, to enhance the automation, in this study, the author has proposed the merging of all the models in a calibrated manner. Giving models' probabilities to another machine learning model to be trained on the models' decision would help to make the algorithm decide which model gets higher weight in the decision-making, and the top layer model will identify the best separation point (threshold).

In general, the spammers are always ahead in inventing ways to bypass filter and detection systems, so researchers' duty is to reduce the gap. Reducing the gap means fewer victims, which leads to less profit for spammers. The main concept of SuspectRate is being dynamic, as spammers do not rely on one method or one source of data. There is a need to build a system that periodically retrains its models and evaluates features and characteristics so that it eliminates less-effective ones. Having different types of models will make it difficult for spammers to fool all models, each of which could have different priorities of features and diverse building methods.

7.2 Research limitation

The process of collecting a high-precision training dataset with heavyweight features is expensive in terms of the computing resources required. A manual labelling process can be used to increase the quality of the training data, however typically this leads to the reduction of the size of the available data. Obtaining more data may produce

a more accurate and efficient model. Due to the time constraints, the author has not been able to explore all promising feature options, such as web page screenshot images. Furthermore, the researchers could not try promising new algorithms, such as deep learning classification.

7.3 Future work

Several tools could be deployed that researchers can use in the context of building machine learning-based spam detection systems. First, a deep learning model could help in reducing the overhead of selecting features, as deep learning models have the ability to internally give higher weights to good features and low weights to unimportant ones. Therefore, researchers could focus more on optimising the model hyper-parameters and model structures of deep learning models. Regarding the hyper-parameters, researchers could use the Bayesian optimisation algorithm to optimise the selection of tuning hyper-parameters.

This research opens up further research opportunities to increase detection system performance, scalability and usability. One of the never-ending challenges for internet security domain expert is coming up with new and discriminative features. As lightweight features are weakened by spammers finding ways to overcome this, researchers are seeking more in-depth features which can be derived from web pages and their attached files and images. The system can store web pages' screenshots which can be used in the future. Applying deep learning-based models to features of this type could show improvement, as deep learning algorithms have proven effective on visual data.

Another research opportunity is enhancing the system's scalability, as the rapid spread of content in social networks requires a scalable system that can analyse content

and make a decision in real time or near to real time. Therefore, there is a need to build the system on a scalable environment such as a serverless environment. Moreover, there is an opportunity to apply this research to do certain users content assessment to detect spam fake accounts.

APPENDIX A

Example of a Tweet JSON data sample

```
{
  "created_at": "Sun Apr 29 17:55:11 +0000 2018",
  "id": 990650721220603904,
  "id_str": "990650721220603904",
  "text": "It was amazing experience #PyDataLDN, great talks and meet new friends. Well done @pydatalondon https://t.co/oqRIDYelDO",
  "truncated": false,
  "entities": {
    "hashtags": [
      {
        "text": "PyDataLDN",
        "indices": [ 26, 36 ]
      }
    ],
    "symbols": [ ],
    "user_mentions": [
      {
        "screen_name": "pydatalondon",
        "name": "PyData London",
        "id": 2431816790,
        "id_str": "2431816790",
        "indices": [ 82, 95 ]
      }
    ],
    "urls": [ ],
    "media": [
      {
        "id": 990647145345806336,
        "id_str": "990647145345806336",
        "indices": [ 96, 119 ],
        "media_url": "http://pbs.twimg.com/media/Db98VHwXcAA7xGm.jpg",
        "media_url_https": "https://pbs.twimg.com/media/Db98VHwXcAA7xGm.jpg",
        "url": "https://t.co/oqRIDYelDO",
        "display_url": "pic.twitter.com/oqRIDYelDO",
        "expanded_url": "https://twitter.com/mhdfadhil/status/990650721220603904/photo/1",

```



```

    "type": "photo",
    "sizes": {
      "thumb": { "w": 150, "h": 150, "resize": "crop" },
      "small": { "w": 403, "h": 680, "resize": "fit" },
      "large": { "w": 1213, "h": 2048, "resize": "fit" },
      "medium": { "w": 711, "h": 1200, "resize": "fit" }
    }
  }
],
},
"extended_entities": {
  "media": [
    {
      "id": 990647145345806336,
      "id_str": "990647145345806336",
      "indices": [96, 119],

"media_url": "http://pbs.twimg.com/media/Db98VHwXcAA7xGm.jpg",

"media_url_https": "https://pbs.twimg.com/media/Db98VHwXcAA7xGm.jpg",
      "url": "https://t.co/oqRlDYelDO",
      "display_url": "pic.twitter.com/oqRlDYelDO",

"expanded_url": "https://twitter.com/mhdfadhil/status/990650721220603904/photo/1",
      "type": "photo",
      "sizes": {
        "thumb": { "w": 150, "h": 150, "resize": "crop" },
        "small": { "w": 403, "h": 680, "resize": "fit" },
        "large": { "w": 1213, "h": 2048, "resize": "fit" },
        "medium": { "w": 711, "h": 1200, "resize": "fit" }
      }
    },
    {
      "id": 990650691621355520,
      "id_str": "990650691621355520",
      "indices": [ 96, 119],
      "media_url": "http://pbs.twimg.com/media/Db9_jiqWAAArgNb.jpg",

"media_url_https": "https://pbs.twimg.com/media/Db9_jiqWAAArgNb.jpg",
      "url": "https://t.co/oqRlDYelDO",
      "display_url": "pic.twitter.com/oqRlDYelDO",

"expanded_url": "https://twitter.com/mhdfadhil/status/990650721220603904/photo/1",
      "type": "photo",
      "sizes": {
        "large": { "w": 1040, "h": 780, "resize": "fit" },

```

```

        "thumb":{"w":150, "h":150, "resize":"crop" },
        "small":{"w":680, "h":510, "resize":"fit" },
        "medium":{"w":1040, "h":780, "resize":"fit" }
    }
}
],
},
"source": "<a href=\\\"http://twitter.com/download/android\\\"
rel=\\\"nofollow\\\">Twitter for Android</a>",
"in_reply_to_status_id":null,
"in_reply_to_status_id_str":null,
"in_reply_to_user_id":null,
"in_reply_to_user_id_str":null,
"in_reply_to_screen_name":null,
"user":{
    "id":104074179,
    "id_str":"104074179",
    "name":"Mohammed Fadhil",
    "screen_name":"mhd fadhil",
    "location":"Stoke-on-Trent, England",
"description":"PhD student at Keele university, Interested in Web scraping, Machine
Learning and internet security.",
"url":"https://t.co/92yeKvCHIA",
"entities":{
    "url":{
        "urls":[
            {
                "url":"https://t.co/92yeKvCHIA",
                "expanded_url":"http://www.scm.keele.ac.uk/staff/m_al-janabi/",
                "display_url":"scm.keele.ac.uk/staff/m_al-jan\\u2026",
                "indices":[ 0, 23 ]
            }
        ]
    },
    "description":{
        "urls":[ ]
    }
},
"protected":false,
"followers_count":353,
"friends_count":859,
"listed_count":17,
"created_at":"Tue Jan 12 04:59:33 +0000 2010",
"favourites_count":484,
"utc_offset":null,
"time_zone":null,
"geo_enabled":true,

```

```

    "verified":false,
    "statuses_count":215,
    "lang":"en",
    "contributors_enabled":false,
    "is_translator":false,
    "is_translation_enabled":false,
    "profile_background_color":"1A1B1F",
    "profile_background_image_url":"http://abs.twimg.com/images/themes/theme9/bg.gif",
    "profile_background_image_url_https":"https://abs.twimg.com/images/themes/theme9/bg.gif",
    "profile_background_tile":false,
    "profile_image_url":"http://pbs.twimg.com/profile_images/696827523615821824/UwFpZ9Fp_normal.jpg",
    "profile_image_url_https":"https://pbs.twimg.com/profile_images/696827523615821824/UwFpZ9Fp_normal.jpg",
    "profile_banner_url":"https://pbs.twimg.com/profile_banners/104074179/1413384703",
    "profile_link_color":"2FC2EF",
    "profile_sidebar_border_color":"181A1E",
    "profile_sidebar_fill_color":"252429",
    "profile_text_color":"666666",
    "profile_use_background_image":true,
    "has_extended_profile":true,
    "default_profile":false,
    "default_profile_image":false,
    "following":false,
    "follow_request_sent":false,
    "notifications":false,
    "translator_type":"none"
  },
  "geo":null,
  "coordinates":null,
  "place":null,
  "contributors":null,
  "is_quote_status":false,
  "retweet_count":2,
  "favorite_count":9,
  "favorited":false,
  "retweeted":false,
  "possibly_sensitive":false,
  "possibly_sensitive_appealable":false,
  "lang":"en"}

```

APPENDIX B

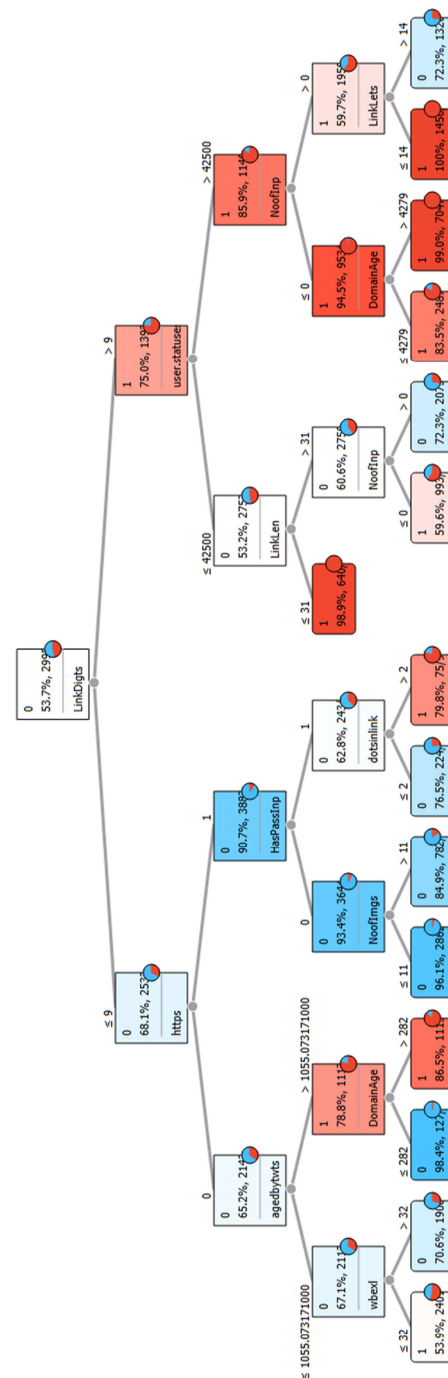
Python libraries used in building the system

adblockparser	0.7
beautifulsoup4	4.6.0
datefinder	0.6.0
dateparser	0.6.0
eli5	0.8
flask-bootstrap	3.3.7.1
flask-cors	3.0.2
flask-mysqldb	0.2.0
flask-wtf	0.14.2
flask	0.12
matplotlib	2.0.0
mlxtend	0.11.0
mysqlclient	1.3.12
nltk	3.2.2
numpy	1.14.3
pandas-profiling	1.4.1
pandas	0.23.1
pip	10.0.1
plotly	2.7.0
pymongo	3.4.0
requests-file	1.4.1
requests-oauthlib	0.8.0

requests	2.12.4
scikit-learn	0.18.1
scipy	1.1.0
seaborn	0.7.1
selenium-requests	1.3
selenium	3.4.3
spyder	3.2.4
tldextract	2.0.2
tweepy	3.5.0
wtforms	2.1
xgboost	0.6
xlsxwriter	0.9.6

APPENDIX C

Example of a visualised tree model



REFERENCES

- [1] Twitter Inc., “Twitter usage / Company Facts,” 2016. [Online]. Available: <https://about.twitter.com/company>. [Accessed: 03-Jun-2015].
- [2] M. Fire, R. Goldschmidt, and Y. Elovici, “Online Social Networks: Threats and Solutions Survey,” *IEEE Commun. Surv. TUTORIALS Online*, vol. 16, no. 4, pp. 1–20, 2013.
- [3] J. Jang-Jaccard and S. Nepal, “A survey of emerging threats in cybersecurity,” *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, Aug. 2014.
- [4] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, “The rise of social bots,” *Commun. ACM*, vol. 59, no. 7, pp. 96–104, Jun. 2016.
- [5] M. B. Zafar, P. Bhattacharya, N. Ganguly, K. P. Gummadi, and S. Ghosh, “Sampling Content from Online Social Networks,” *ACM Trans. Web*, vol. 9, no. 3, pp. 1–33, 2015.
- [6] Networked Insights, “How Dirty is Big Data ?,” 2015. [Online]. Available: <http://info.networkedinsights.com/Dirty-Data-LP.html>. [Accessed: 02-Jun-2015].
- [7] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “The socialbot network,” in *Proceedings of the 27th Annual Computer Security Applications Conference on - ACSAC '11*, 2011, p. 93.
- [8] L. Bilge and T. Dumitras, “Before We Knew It,” *Proc. 19th ACM Conf. Comput. Commun. Secur. - CCS '12*, p. 833, 2012.
- [9] X. Zheng, Z. Zeng, Z. Chen, Y. Yu, and C. Rong, “Detecting spammers on social

- networks,” *Neurocomputing*, vol. 159, no. 0, pp. 27–34, Jul. 2015.
- [10] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, “Aiding the detection of fake accounts in large scale social online services,” in *NSDI’12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, p. 15.
 - [11] R. Wolf, “Redefining the concept of insulation,” *Technical Textiles International*, 1996. [Online]. Available: <http://nexgate.com/wp-content/uploads/2013/09/Nexgate-2013-State-of-Social-Media-Spam-Research-Report.pdf>. [Accessed: 28-Apr-2015].
 - [12] C. Grier, K. Thomas, V. Paxson, and M. Zhang, “@spam: The Underground on 140 Characters or Less,” in *Proceedings of the 17th ACM conference on Computer and communications security - CCS ’10*, 2010, p. 27.
 - [13] J. Echeverria and S. Zhou, “Discovery, Retrieval, and Analysis of the ‘Star Wars’ Botnet in Twitter,” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017 - ASONAM ’17*, 2017, pp. 1–8.
 - [14] K. Lee, J. Caverlee, K. Y. Kamath, and Z. Cheng, “Detecting collective attention spam,” in *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality - WebQuality ’12*, 2012, p. 48.
 - [15] K. Thomas, “The Role of the Underground Economy in Social Network Spam and Abuse,” University of California at Berkeley, 2013.
 - [16] J. Cao, Q. Fu, Q. Li, and D. Guo, “Discovering Hidden Suspicious Accounts in Online Social Networks,” *Inf. Sci. (Ny)*, vol. 394–395, pp. 123–140, 2017.
 - [17] P. R. Badri Satya, B. Satya, K. Lee, D. Lee, and J. J. Zhang, “Uncovering Fake

- Likers in Online Social Networks,” *ACM Trans. Internet Technol.*, 2016.
- [18] A. Tewari, A. K. Jain, and B. B. Gupta, “Recent survey of various defense mechanisms against phishing attacks,” *J. Inf. Priv. Secur.*, vol. 12, no. 1, pp. 3–13, 2016.
 - [19] Y.-R. Lin, D. Margolin, B. Keegan, A. Baronchelli, and D. Lazer, “#Bigbirds Never Die: Understanding Social Dynamics of Emergent Hashtag,” *Proc. 7th Int. AAAI Conf. Weblogs Soc. Media (ICWSM 2013)*, vol. 8, no. Disasters, Mar. 2013.
 - [20] S. Stieglitz and L. Dang-Xuan, “Social media and political communication: a social media analytics framework,” *Soc. Netw. Anal. Min.*, vol. 3, no. 4, pp. 1277–1291, Dec. 2013.
 - [21] N. Azman, “Dark Retweets: An Investigation of Non-Conventional Retweeting Patterns,” UNIVERSITY OF SOUTHAMPTON, 2014.
 - [22] X. Zhang, S. Zhu, and W. Liang, “Detecting spam and promoting campaigns in the Twitter social network,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 1194–1199, 2012.
 - [23] M. Verma, D. Divya, and S. Sofat, “Techniques to Detect Spammers in Twitter-A Survey,” *Int. J. Comput. Appl.*, vol. 85, no. 10, pp. 27–32, Jan. 2014.
 - [24] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, “Detection and Resolution of Rumours in Social Media: A Survey,” vol. 51, no. 2, 2017.
 - [25] H. Allcott and M. Gentzkow, “Social Media and Fake News in the 2016 Election,” *J. Econ. Perspect.*, vol. 31, no. 2, pp. 211–236, 2017.
 - [26] S. Stieglitz, F. Brachten, D. Berthel, and M. Schlaus, “Social Computing and Social Media. Human Behavior,” in *Social Computing and Social Media*.

- Human Behavior*, 2017, vol. 10282, no. December, pp. 379–395.
- [27] N. Nikiforakis, F. Maggi, G. Stringhini, M. Z. Rafique, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, and S. Zanero, “Stranger Danger: Exploring the Ecosystem of Ad-based URL Shortening Services,” in *Proceedings of the 23rd international conference on World wide web - WWW '14*, 2014, pp. 51–62.
 - [28] A. Almaatouq, E. Shmueli, M. Nouh, A. Alabdulkareem, V. K. Singh, M. Alsaleh, A. Alarifi, A. Alfari, and A. ‘Sandy’ Pentland, “If it looks like a spammer and behaves like a spammer, it must be a spammer: analysis and detection of microblogging spam accounts,” *Int. J. Inf. Secur.*, vol. 15, no. 5, pp. 475–491, 2016.
 - [29] Webopedia.com, “What is AAA? A Webopedia Definition,” 2015. [Online]. Available: <http://www.webopedia.com/TERM/A/AAA.html>. [Accessed: 03-Jun-2015].
 - [30] R. a. Haraty and S. Massalkhy, *Security and Privacy Preserving in Social Networks*. Springer Science & Business Media, 2013.
 - [31] E. Zangerle and G. Specht, ““Sorry, I was hacked,”” in *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*, 2014, pp. 587–593.
 - [32] K. Thomas, F. Li, C. Grier, and V. Paxson, “Consequences of Connectivity: Characterizing Account Hijacking on Twitter,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 489–500.
 - [33] E. Protalinski, “Facebook estimates that between 5.5% and 11.2% of accounts are fake,” *The Next Web*, 2014. [Online]. Available: <http://tnw.to/b4ZyU>. [Accessed: 03-Jun-2015].

- [34] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “The Paradigm-Shift of Social Spambots,” in *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, 2017, pp. 963–972.
- [35] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Who is tweeting on Twitter,” in *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC '10*, 2010, p. 21.
- [36] Z. Chu, I. Widjaja, and H. Wang, “Detecting social spam campaigns on Twitter,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7341 LNCS, pp. 455–472, 2012.
- [37] C. Cao and J. Caverlee, “Detecting Spam URLs in Social Media via Behavioral Analysis,” in *Advances in Information Retrieval*, vol. 9022, A. Hanbury, G. Kazai, A. Rauber, and N. Fuhr, Eds. Springer International Publishing, 2015, pp. 703–714.
- [38] D. Bock Clark, “The Bot Bubble: How Click Farms Have Infiltrated Social Media Currency,” 2015. [Online]. Available: <http://www.newrepublic.com/article/121551/bot-bubble-click-farms-have-inflated-social-media-currency>. [Accessed: 03-Jun-2015].
- [39] T. N. Jagatic, N. A. Johnson, M. Jakobsson, F. Menczer, B. T. N. Jagatic, N. A. Johnson, and M. Jakobsson, “SOCIAL PHISHING.,” *Commun. ACM*, vol. 50, no. 10, pp. 94–100, Oct. 2007.
- [40] F. Klien and M. Strohmaier, “Short Links Under Attack: Geographical Analysis of Spam in a URL Shortener Network,” in *Proceedings of the 23rd ACM conference on Hypertext and social media - HT '12*, 2012, p. 83.

- [41] D. S. Silnov, "An analysis of modern approaches to the delivery of unwanted emails (spam)," *Indian J. Sci. Technol.*, vol. 9, no. 4, pp. 1–4, 2016.
- [42] P. Ponce-Cruz and F. D. Ramírez-Figueroa, *Intelligent Control Systems with LabVIEWTM*. London: Springer London, 2010.
- [43] S. Wen, Z. Zhao, and H. Yan, "Detecting Malicious Websites in Depth through Analyzing Topics and Web-pages," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy - ICCSP 2018*, 2018, pp. 128–133.
- [44] C. Chen, J. Zhang, Y. Xiang, W. Zhou, and J. Oliver, "Spammers Are Becoming 'Smarter' on Twitter," *IT Prof.*, vol. 18, no. 2, pp. 66–70, Mar. 2016.
- [45] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Towards Detecting Compromised Accounts on Social Networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 14, no. 4, pp. 447–460, Jul. 2017.
- [46] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering Social Network Sybils in the Wild," *ACM Trans. Knowl. Discov. Data*, vol. 8, no. 1, p. 2:1–2:29, 2014.
- [47] P. Gao, N. Z. Gong, S. Kulkarni, K. Thomas, and P. Mittal, "SybilFrame: A Defense-in-Depth Framework for Structure-Based Sybil Detection," *Comput. Res. Repos.*, p. 17, 2015.
- [48] I. A. Bara, C. J. Fung, and T. Dinh, "Enhancing Twitter spam accounts discovery using cross-account pattern mining," *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*. pp. 491–496, 2015.
- [49] K. Tretyakov, "Machine Learning Techniques in Spam Filtering," *Data Min.*

- Probl. Semin. MTAT.03.177*, no. May, pp. 60–79, 2004.
- [50] Y. Zhang and Y. Tong, “Mining trust relationships from online social networks,” *J. Comput. Sci. Technol.*, vol. 27, no. 3, pp. 492–505, 2012.
 - [51] P.-A. Vervier and O. Thonnard, “SpamTracer: How stealthy are spammers?,” in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 3477–3482.
 - [52] A. Nakulas, L. Ekonomou, S. Kourtesi, G. P. Fotis, and E. Zoulias, “A review of techniques to counter spam and spit,” in *Lecture Notes in Electrical Engineering*, vol. 27 LNEE, no. VOL.1, Springer, 2009, pp. 501–510.
 - [53] S. K. Dehade and A. M. Bagade, “A Review on Detecting Automation on Twitter Accounts,” *Eur. J. Adv. Eng. Technol.*, vol. 2, no. 2, pp. 69–72, 2015.
 - [54] Chris Richardson, “Google Discusses Its Safe Browsing Record | WebProNews.” [Online]. Available: <http://www.webpronews.com/google-discusses-its-safe-browsing-record-2012-06>. [Accessed: 03-Jun-2015].
 - [55] G. Developers, “Safe Browsing API,” *Developers.Google.Com*, 2015. [Online]. Available: <https://developers.google.com/safe-browsing/?hl=en>. [Accessed: 03-Jun-2015].
 - [56] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, “Design and evaluation of a real-time URL spam filtering service,” in *Proceedings - IEEE Symposium on Security and Privacy*, 2011, pp. 447–462.
 - [57] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An Empirical Analysis of Phishing Blacklists,” in *6th Conference on Email and Anti-Spam*, 2009.
 - [58] M. Al-janabi, E. De Quincey, and P. Andras, “Using supervised machine learning algorithms to detect suspicious URLs in online social networks,” in

- Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 1104–1111.
- [59] J. Tang, Y. Chang, and H. Liu, “Mining Social Media with Social Theories: A Survey,” *SIGKDD Explor. Newsl*, vol. 15, no. 1, pp. 20–29, 2014.
 - [60] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
 - [61] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc, 2007.
 - [62] S. Yoo, Y. Yang, F. Lin, C. Moon, S. Acemoglu, and S. Acemoglu, “Mining social networks for personalized email prioritization,” *15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, KDD '09*, pp. 967–975, 2009.
 - [63] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: A review of classification and combining techniques,” *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, 2006.
 - [64] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, “Detecting spammers on twitter,” *Collab. Electron. Messag. anti-abuse spam Conf.*, vol. 6, p. 12, 2010.
 - [65] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed., vol. 43. Beijing: O’Reilly, 2009.
 - [66] F. Maiorana, “Feature Selection with Kohonen Self Organizing Classification Algorithm,” *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 2, no. 9, pp. 47–52, 2008.
 - [67] G. Forman, “An Extensive Empirical Study of Feature Selection Metrics for Text Classification,” *CrossRef List. Deleted DOIs*, vol. 1, no. 7–8, pp. 1289–1305, 2000.

- [68] L. Zhang, J. Zhu, and T. Yao, “An evaluation of statistical spam filtering techniques,” *ACM Trans. Asian Lang. Inf. Process.*, vol. 3, no. 4, pp. 243–269, Dec. 2004.
- [69] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer New York, 2009.
- [70] T. S. Guzella and W. M. Caminhas, “A review of machine learning approaches to Spam filtering,” *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, Sep. 2009.
- [71] R. Matsumoto, “Some empirical results on two spam detection methods,” *Proc. 2004 IEEE Int. Conf. Inf. Reuse Integr. 2004. IRI 2004.*, pp. 198–203, 2004.
- [72] V. Zorkadis, D. A. Karras, and M. Panayotou, “Efficient information theoretic strategies for classifier combination, feature extraction and performance evaluation in improving false positives and false negatives for spam e-mail filtering,” *Neural Networks*, vol. 18, no. 5–6, pp. 799–807, Jan. 2005.
- [73] A. Liaw, M. Wiener, and J. Hebebrand, “Classification and regression by randomForest,” *R news*, vol. 2, no. 3, pp. 18–22, 01-Dec-2002.
- [74] V. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake, “Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5528, pp. 447–456, 2009.
- [75] M. Alsaleh, A. Alarifi, A. M. Al-Salman, M. Alfayez, and A. Almuhaysin, “TSD: Detecting sybil accounts in twitter,” *Proc. - 2014 13th Int. Conf. Mach. Learn. Appl. ICMLA 2014*, pp. 463–469, 2014.
- [76] T. G. Dietterich, “Ensemble Methods in Machine Learning,” *MCS '00 Proc.*

- First Int. Work. Mult. Classif. Syst.*, pp. 1–15, 2000.
- [77] S. Hara and K. Hayashi, “Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach,” in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018, vol. 84, pp. 77–85.
 - [78] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD '16*, pp. 785–794, 2016.
 - [79] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
 - [80] M. Pal, “Random forest classifier for remote sensing classification,” *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, Jan. 2005.
 - [81] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.
 - [82] J. R. Quinlan, “Bagging, boosting, and C4.5,” *Proc. Thirteen. Natl. Conf. Artif. Intell.*, vol. 5, no. Quinlan 1993, pp. 725–730, 2006.
 - [83] L. Rokach, “Ensemble-based classifiers,” *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010.
 - [84] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, “Towards Detecting Anomalous User Behavior in Online Social Networks,” in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 223–238.
 - [85] S. Dinh, T. Azeb, F. Fortin, D. Mouheb, and M. Debbabi, “Spam campaign detection, analysis, and investigation,” *Digit. Investig.*, vol. 12, no. S1, pp. S12–S21, 2015.
 - [86] C. Chen, J. Zhang, X. Chen, Y. Xiang, and W. Zhou, “6 million spam tweets: A

- large ground truth for timely Twitter spam detection,” *IEEE Int. Conf. Commun.*, vol. 2015–Septe, pp. 7065–7070, 2015.
- [87] N. Gupta, A. Aggarwal, and P. Kumaraguru, “Bit.ly/malicious: Deep dive into short URL based e-crime detection,” *eCrime Researchers Summit, eCrime*, vol. 2014–Janua. pp. 14–24, 2014.
- [88] T. Wu, S. Wen, S. Liu, J. Zhang, Y. Xiang, M. Alrubaian, and M. M. Hassan, “Detecting spamming activities in twitter based on deep-learning technique,” *Concurr. Comput.*, vol. 29, no. 19, pp. 1–11, 2017.
- [89] S. Liu, J. Zhang, and Y. Xiang, “Statistical Detection of Online Drifting Twitter Spam,” *Proc. 11th ACM Asia Conf. Comput. Commun. Secur. - ASIA CCS '16*, pp. 1–10, 2016.
- [90] B. Wang, A. Zubiaga, M. Liakata, and R. Procter, “Making the most of tweet-inherent features for social spam detection on twitter,” *CEUR Workshop Proc.*, vol. 1395, pp. 10–16, 2015.
- [91] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru, “PhishAri: Automatic realtime phishing detection on twitter,” *eCrime Res. Summit, eCrime*, pp. 1–12, 2012.
- [92] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, “Twitter spammer detection using data stream clustering,” *Inf. Sci. (Ny)*, vol. 260, pp. 64–73, Mar. 2014.
- [93] S. Lee and J. Kim, “Warning bird: A near real-time detection system for suspicious URLs in twitter stream,” *IEEE Trans. Dependable Secur. Comput.*, vol. 10, no. 3, pp. 183–195, 2013.
- [94] S. J. Soman and S. Murugappan, “Detecting malicious tweets in trending topics

- using clustering and classification,” in *2014 International Conference on Recent Trends in Information Technology, ICRTIT 2014*, 2014, pp. 1–6.
- [95] K. Lee, J. Caverlee, and S. Webb, “Uncovering social spammers: social honeypots + machine learning,” *SIGIR’10, July 19–23, 2010, Geneva, Switz.*, no. i, pp. 435–442, 2010.
- [96] A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu, and C. Yang, “CATS: Characterizing automation of Twitter spammers,” *2013 5th International Conference on Communication Systems and Networks, COMSNETS 2013*. pp. 1–10, 2013.
- [97] A. Valdes, A. Valdes, K. Skinner, and K. Skinner, *Recent Advances in Intrusion Detection*, vol. 1907, no. October. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.
- [98] F. Li, M. Huang, Y. Yang, and X. Zhu, “Learning to identify review spam,” in *Proceedings of the twenty-second international joint conference on artificial intelligence*, 2011, pp. 2488–2493.
- [99] W. Guan, H. Gao, M. Yang, Y. Li, H. Ma, W. Qian, Z. Cao, and X. Yang, “Analyzing user behavior of the micro-blogging website Sina Weibo during hot social events,” *Phys. A Stat. Mech. its Appl.*, vol. 395, pp. 340–351, Feb. 2014.
- [100] Twitter, “The Streaming APIs | Twitter Developers,” 2015. .
- [101] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Chudhary, “Towards Online Spam Filtering in Social Networks,” in *Network and Distributed System Security Symposium*, 2012, pp. 1–16.
- [102] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, “Detecting and characterizing social spam campaigns,” in *Proceedings of the 10th ACM*

- SIGCOMM conference on Internet measurement*, 2010, pp. 35–47.
- [103] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Fame for sale: Efficient detection of fake Twitter followers,” *Decis. Support Syst.*, vol. 80, no. July 2012, pp. 56–71, 2015.
 - [104] M. McCord and M. Chuah, “Spam detection on twitter using traditional classifiers,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6906 LNCS, pp. 175–186, 2011.
 - [105] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?,” *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 6, pp. 811–824, 2012.
 - [106] H. Shen and X. Liu, “Detecting Spammers on Twitter Based on Content and Social Interaction,” *Proc. - 2015 Int. Conf. Netw. Inf. Syst. Comput. ICNISC 2015*, pp. 413–417, 2015.
 - [107] S. Sedhai and A. Sun, “HSpam14: A Collection of 14 Million Tweets for Hashtag-Oriented Spam Research,” *Proc. 38th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '15*, pp. 223–232, 2015.
 - [108] E. M. Clark, J. R. Williams, C. A. Jones, R. A. Galbraith, C. M. Danforth, and P. S. Dodds, “Sifting robotic from organic text: A natural language approach for detecting automation on Twitter,” *J. Comput. Sci.*, vol. 16, pp. 1–7, 2016.
 - [109] S. Stieglitz, F. Brachten, D. Berthel, and M. Schlaus, “Social Computing and Social Media. Human Behavior,” in *Social Computing and Social Media. Human Behavior*, 2017, vol. 10282, no. December, pp. 379–395.
 - [110] C. Whittaker, B. Ryner, and M. Nazif, “Large-Scale Automatic Classification of Phishing Pages,” in *Ndss '10*, 2010, vol. 10.

- [111] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," *Proc. 3rd ACM Work. Artif. Intell. Secur. - AISec '10*, no. August 2016, p. 54, 2010.
- [112] M. Khonji, A. Jones, and Y. Iraqi, "A novel Phishing classification based on URL features," *2011 IEEE GCC Conference and Exhibition, GCC 2011*. pp. 221–224, 2011.
- [113] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th international conference on Security and privacy in communication networks - SecureComm '08*, 2008, p. 1.
- [114] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," *Spec. Interes. tracks posters 14th Int. Conf. World Wide Web - WWW '05*, no. September 2015, p. 1060, 2005.
- [115] P. Burnap, A. Javed, O. F. Rana, and M. S. Awan, "Real-time classification of malicious URLs on Twitter using machine activity data," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 2015, pp. 970–977.
- [116] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to Spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, 2009.
- [117] A. D'Ambrogio, "Computer and Information Sciences - ISCIS 2005," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3733, pp. 371–381, 2005.
- [118] M. Singh, D. Bansal, and S. Sofat, "Detecting Malicious Users in Twitter using Classifiers," *7th Int. Conf. Secur. Inf. Networks*, p. 247, 2014.
- [119] A. B. M. Moniruzzaman and S. A. Hossain, "Nosql database: New era of

- databases for big data analytics-classification, characteristics and comparison,” *arXiv Prepr. arXiv1307.0191*, vol. 6, no. 4, pp. 1–14, 2013.
- [120] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.-J. Lee, M. Mellia, C. Kruegel, and G. Vigna, “Nazca: Detecting Malware Distribution in Large-Scale Networks,” *Netw. Distrib. Syst. Secur. Symp.*, pp. 1–16, 2014.
- [121] D. Wang, S. Navathe, L. Liu, D. Irani, A. Tamersoy, and C. Pu, “Click Traffic Analysis of Short URL Spam on Twitter,” *Proc. 9th IEEE Int. Conf. Collab. Comput. Networking, Appl. Work.*, pp. 250–259, 2013.
- [122] K. Thomas, C. Grier, D. Song, and V. Paxson, “Suspended accounts in retrospect: an analysis of twitter spam,” in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, 2011, pp. 243–258.
- [123] S. Gupta, A. Khattar, A. Gogia, P. Kumaraguru, and T. Chakraborty, “Collective Classification of Spam Campaigners on Twitter: A Hierarchical Meta-Path Based Approach,” 2018.
- [124] N. Chavoshi, H. Hamooni, and A. Mueen, “Identifying Correlated Bots in Twitter,” in *Social Informatics*, 2016, vol. 10046, no. November, pp. 14–21.
- [125] S. Liu, Y. Wang, J. Zhang, C. Chen, and Y. Xiang, “Addressing the class imbalance problem in Twitter spam detection using ensemble learning,” *Comput. Secur.*, vol. 69, pp. 35–49, 2017.
- [126] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, “Survey of review spam detection using machine learning techniques,” *J. Big Data*, vol. 2, no. 1, 2015.
- [127] T. K. Ho, J. J. Hull, and S. N. Srihari, “Decision Combination in Multiple

- Classifier Systems,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, 1994.
- [128] W. Herzallah, H. Faris, and O. Adwan, “Feature engineering for detecting spammers on Twitter: Modelling and analysis,” *J. Inf. Sci.*, p. 016555151668429, 2017.
- [129] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, “Two years of short URLs internet measurement: security threats and countermeasures,” *Proc. 22nd Int. Conf. World Wide Web*, pp. 861–872, 2013.
- [130] K. Krol, M. Moroz, and M. A. Sasse, “Don’t work. Can’t work? Why it’s time to rethink security warnings,” *2012 7th Int. Conf. Risks Secur. Internet Syst.*, pp. 1–8, 2012.
- [131] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, “The Dark Alleys of Madison Avenue,” *Proc. 2014 Conf. Internet Meas. Conf. - IMC ’14*, pp. 373–380, 2014.
- [132] C. C. Aggarwal, “Opinion Mining and Sentiment Analysis,” in *Machine Learning for Text*, Cham: Springer International Publishing, 2018, pp. 413–434.
- [133] S. Saumya and J. P. Singh, “Detection of spam reviews: a sentiment analysis approach,” *CSI Trans. ICT*, vol. 6, no. 2, pp. 137–148, 2018.
- [134] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory Undersampling for Class Imbalance Learning,” *IEEE Trans. Syst. Man Cybern.*, vol. 39, no. 2, pp. 539–550, 2009.
- [135] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press, 2013.
- [136] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27,

- no. 8, pp. 861–874, 2006.
- [137] G. M. Tavares and G. M. Tavares, “User Classification on Online Social Networks by Post Frequency User Classification on Online Social Networks by Post Frequency,” no. June, 2017.
- [138] C. Yang, R. C. Harkreader, and G. Gu, “Empirical evaluation and new design for fighting evolving twitter spammers,” *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1280–1293, 2013.
- [139] J. C. Ross and P. E. Allen, “Random Forest for improved analysis efficiency in passive acoustic monitoring,” *Ecol. Inform.*, vol. 21, pp. 34–39, 2014.
- [140] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, “A comparison of decision tree ensemble creation techniques,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, 2007.
- [141] C. A. Provan, L. Cook, and J. Cunningham, “A probabilistic airport capacity model for improved ground delay program planning,” *AIAA/IEEE Digit. Avion. Syst. Conf. - Proc.*, pp. 1–12, 2011.
- [142] J. P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. E. Brodley, “Pruning decision trees with misclassification costs,” in *Machine Learning: ECML-98: 10th European Conference on Machine Learning*, vol. 1398, C. Nédellec and C. Rouveirol, Eds. Chemnitz, Germany: Springer Berlin Heidelberg, 1998, pp. 131–136.
- [143] Q. Xu, E. W. Xiang, Q. Yang, J. Du, and J. Zhong, “SMS Spam Detection Using Noncontent Features,” *IEEE Intell. Syst.*, vol. 27, no. 6, pp. 44–51, Nov. 2012.
- [144] M. Dash, H. Liu, M. Dash ', and H. Liu, “Feature selection for classification,” *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, 1997.

- [145] Y. Zhang, S. Wang, P. Phillips, and G. Ji, “Binary PSO with mutation operator for feature selection using decision tree applied to spam detection,” *Knowledge-Based Syst.*, vol. 64, pp. 22–31, 2014.
- [146] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, “Understanding variable importances in forests of randomized trees,” *Adv. Neural Inf. Process. Syst.* 26, pp. 431–439, 2013.
- [147] X. Zhang, S. Zhu, and W. Liang, “Detecting spam and promoting campaigns in the Twitter social network,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, vol. 10, no. 1, pp. 1194–1199, 2012.
- [148] R. Zafarani and H. Liu, “10 Bits of Surprise,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15*, 2015, pp. 423–431.
- [149] P. Domingos, “A few useful things to know about machine learning,” *Commun. ACM*, vol. 55, no. 10, p. 78, Oct. 2012.
- [150] T. G. Dietterich, “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees,” *Mach. Learn.*, vol. 40, pp. 139–157, 2000.
- [151] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, “Tackling the problem of classification with noisy data using Multiple Classifier Systems: Analysis of the performance and robustness,” *Inf. Sci. (Ny)*, vol. 247, pp. 1–20, 2013.
- [152] S. B. Kotsiantis and P. E. Pintelas, “Combining Bagging and Boosting,” *Comput. Intell.*, vol. 1, no. 4, pp. 324–333, 2004.